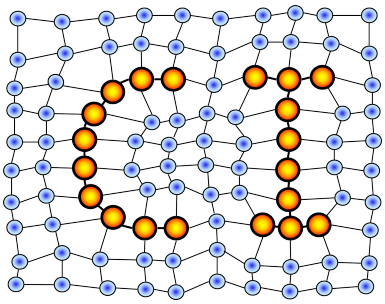

HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Herr
Christian Harth

Erweiterung von Generalized [Relevance|Matrix] Learning Vector Quantization zur Anwendung auf funktionale Daten

2012

Fakultät **Mathematik/Naturwissenschaften/Informatik**

MASTERARBEIT

Erweiterung von Generalized [Relevance|Matrix] Learning Vector Quantization zur Anwendung auf funktionale Daten

Autor:
Christian Harth

Studiengang:
Diskrete und Computerorientierte Mathematik

Seminargruppe:
ZD10

Erstprüfer:
Prof. Dr. Thomas Villmann

Zweitprüfer:
Marika Kästner, M.Sc.

Mittweida, 2012

Bibliografische Angaben

Harth, Christian: Erweiterung von Generalized [Relevance|Matrix] Learning Vector Quantization zur Anwendung auf funktionale Daten, 65 Seiten, 26 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Masterarbeit, 2012

Referat

In dieser Arbeit werden die Verfahren GLVQ und GRLVQ mit der Sobolev-Metrik erweitert und an verschiedene Datensätze mit funktionalen Daten getestet. Außerdem wird ein Ansatz vorgestellt, die Prototypen durch Überlagerungen von Basisfunktionen darzustellen. Dieser Ansatz wird zusätzlich noch auf den GMLVQ angewendet. Hierfür betrachtete man die Gaußfunktionen und Sigmoidfunktionen als Basisfunktionen. Dabei wurden mit der Sobolev-Metrik sehr gute Resultat erzielt.

Danksagung

Ein besonderer Dank gilt meiner Familie und meiner Verlobten, die mich stets motiviert und unterstützt haben. Außerdem danke ich der CIID-Forschungsgruppe, ganz besonders Professor Villmann und Marika Kästner, für die Anregungen und die intensive Betreuung bei meiner Masterarbeit.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungs- und Symbolverzeichnis

Im Folgenden sind die wichtigsten Formelzeichen und Symbole der Arbeit aufgeführt. Hierbei sind Vektoren (Kleinbuchstaben) und Matrizen (Großbuchstaben) fettgedruckt. Skalare hingegen sind normal gedruckt.

Abkürzungsverzeichnis

LVQ	Learning Vector Quantization
GLVQ	Generalized Learning Vector Quantization
GRLVQ	Generalized Relevance Learning Vector Quantization
GLVQ+Sobolev	Generalized Learning Vector Quantization mit der Sobolev-Metrik
GLVQ+Sobolev+Gauß	Generalized Learning Vector Quantization mit der Sobolev-Metrik und der Überlagerung von Gaußfunktionen
GLVQ+Sobolev+Sigmoid	Generalized Learning Vector Quantization mit der Sobolev-Metrik und der Überlagerung von Sigmoidfunktionen
GRLVQ+Sobolev	Generalized Relevance Learning Vector Quantization mit der Sobolev-Metrik
GRLVQ+Sobolev+Gauß	Generalized Relevance Learning Vector Quantization mit der Sobolev-Metrik und der Überlagerung von Gaußfunktionen
GRLVQ+Sobolev+Sigmoid	Generalized Relevance Learning Vector Quantization mit der Sobolev-Metrik und der Überlagerung von Sigmoidfunktionen
GMLVQ	Generalized Matrix Learning Vector Quantization
GMLVQ+Gauß	Generalized Matrix Learning Vector Quantization mit der Überlagerung von Gaußfunktionen
GMLVQ+Sigmoid	Generalized Matrix Learning Vector Quantization mit der Überlagerung von Sigmoidfunktionen

Mathematische Symbole

$\mathbf{x}$, $\mathbf{y}$	Datenvektoren
$\overline{\mathbf{x}}$	geglätteter Datenvektor $\mathbf{x}$
$\mathbf{w}$	Prototyp (Vektor)
$\dot{\mathbf{w}}$	Ableitung erster Ordnung nach t des Prototypen
$\dot{\overline{\mathbf{x}}}$	Ableitung erster Ordnung nach t des geglätteten Prototypen
$\mathbf{w}^+$	Gewinnerprototyp (Vektor)

$\mathbf{w}^-$	Verliererprototyp (Vektor)
k	Klasse
α	Gewicht für den Ableitungsterm in der vereinfachten Sobolev-Metrik
$\boldsymbol{\lambda}$	Relevanzvektor
$\boldsymbol{\Lambda}$	Korrelationsmatrix
$\|\cdot\|_2$, $\|\cdot\|_E$	euklidische Norm
$\|\cdot\|_{\mathscr{L}^p}$	$\mathscr{L}^p$-Norm
$\|\cdot\|_{S(p,R)}$	Sobolev-Norm
$d(\cdot,\cdot)$	Abstandsmaß
$d^+(\mathbf{x})$	Abstand des Gewinnerprototypen zum Datenvektor $\mathbf{x}$
$d^-(\mathbf{x})$	Abstand des Verliererprototypen zum Datenvektor $\mathbf{x}$
$d_{\boldsymbol{\lambda}}(\cdot,\cdot)$	gewichtete euklidische Metrik
$d^+_{\boldsymbol{\lambda}}(\mathbf{x})$	gewichteter euklidischer Abstand des Gewinnerprototypen zum Datenvektor $\mathbf{x}$
$d^-_{\boldsymbol{\lambda}}(\mathbf{x})$	gewichteter euklidischer Abstand des Verliererprototypen zum Datenvektor $\mathbf{x}$
$d_S(\cdot,\cdot,\alpha)$	Vereinfachte Sobolev-Metrik
$d^+_S(\mathbf{x})$	Abstand des Gewinnerprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der Sobolev-Metrik
$d^-_S(\mathbf{x})$	Abstand des Verliererprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der Sobolev-Metrik
$d_{S,\boldsymbol{\lambda}}(\cdot,\cdot,\alpha)$	Vereinfachte Sobolev-Metrik mit Relevanzvektor
$d^+_{S,\boldsymbol{\lambda}}(\mathbf{x})$	Abstand des Gewinnerprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der Sobolev-Metrik mit Relevanzvektor
$d^-_{S,\boldsymbol{\lambda}}(\mathbf{x})$	Abstand des Verliererprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der Sobolev-Metrik mit Relevanzvektor
$d_{\boldsymbol{\Lambda}}(\cdot,\cdot)$	euklidische Metrik mit Korrelationsmatrix $\boldsymbol{\Lambda}$
$d^+_{\boldsymbol{\Lambda}}(\mathbf{x})$	Abstand des Gewinnerprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der euklidischen Metrik mit Korrelationsmatrix
$d^-_{\boldsymbol{\Lambda}}(\mathbf{x})$	Abstand des Verliererprototypen zum Datenvektor $\mathbf{x}$ unter Verwendung der euklidischen Metrik mit Korrelationsmatrix
$\mathscr{X}$	Datenraum
$\mathscr{T}$	Trainingsmenge
$\mathscr{K}$	Klassenmenge
$\mathscr{W}^k$	Menge aller Prototypen für die Klasse k
$\mathscr{B}$	Menge von Basisfunktionen (Basissystem)

$\mathscr{I}$	Indexmenge
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{R}^+$	Menge der positiven reellen Zahlen
$\mathbb{R}^n$	Menge der n-dimensionalen Vektoren mit reellwertigen Komponenten
$\mathscr{C}^1[1,n]$	Menge aller einmal stetig-differenzierbaren Funktionen auf dem Intervall $[1,n]$
κ	Klassifikatorfunktion
r_a	Genauigkeitsrate
μ	Funktion der relativen Fehlerdistanz
$\mu_{\boldsymbol{\lambda}}$	Funktion der relativen Fehlerdistanz unter Verwendung der gewichteten euklidischen Metrik
μ_S	Funktion der relativen Fehlerdistanz unter Verwendung der Sobolev-Metrik
$\mu_{S,\boldsymbol{\lambda}}$	Funktion der relativen Fehlerdistanz unter Verwendung der Sobolev-Metrik mit Relevanzvektor
$\mu_{\boldsymbol{\Lambda}}$	Funktion der relativen Fehlerdistanz unter Verwendung der euklidischen Metrik mit Korrelationsmatrix
E^{GLVQ}	Kostenfunktion des GLVQ
E^{GRLVQ}	Kostenfunktion des GRLVQ
E^{GMLVQ}	Kostenfunktion des GMLVQ
E_S^{GLVQ}	Kostenfunktion des GLVQ unter Verwendung der Sobolev-Metrik
E_S^{GRLVQ}	Kostenfunktion des GRLVQ unter Verwendung der Sobolev-Metrik
b	Basisfunktion
θ_i	Ort der i-ten Basisfunktion
β_i	Gewicht für die i-te Basisfunktion
σ_i	Breite der i-ten Basisfunktion
s_i	Versetzung der i-ten Sigmoidfunktion in y-Richtung
ε	Lernrate für die Prototypen
ε^α	Lernrate für den Parameter α
$\varepsilon^{\boldsymbol{\lambda}}$	Lernrate für den Relevanzvektor λ
$\varepsilon^{\boldsymbol{\Omega}}$	Lernrate für die Korrelationsmatrix $\boldsymbol{\Lambda}$
ε^β	Lernrate für die Höhe der Basisfunktion
ε^σ	Lernrate für die Breite der Basisfunktion
ε^θ	Lernrate für den Ort der Basisfunktion
ε^s	Lernrate für die Versetzung der Basisfunktion in y-Richtung

$\subseteq$ ..... Formelzeichen für Teilmenge

$\forall$ ..... Formelzeichen für den Allquantor (i. W.: „für alle”)

$\dot{\bigcup}$ ..... Formelzeichen für die disjunkte Vereinigung

$\Leftrightarrow$ ..... Formelzeichen für den Ausdruck: „. . .genau dann, wenn. . .”

$\min$ ..... Formelzeichen für den Ausdruck: „Minimum von. . .”

$\frac{\partial \cdot}{\partial \cdot}$ ..... partielle Ableitung von . . . nach . . .

∞ ..... Formelzeichen für „unendlich”

$D^{(j)}\cdot$ ..... Differentialoperator der j-ten Ordnung angewendet auf . . .

π ..... Formelzeichen für die Kreiszahl

1 Einleitung

Ob nun in der Medizin, der Wirtschaft oder in der Technik - das Anwenden von Lernalgorithmen findet zurzeit in der Praxis eine immer größere Beachtung. Diese Algorithmen decken ein großes Spektrum in verschiedensten Anwendungsgebieten ab, wie dem Auswerten von medizinischen Krankheitsbildern oder von Satellitenbildern. In dieser Arbeit wird ein kleiner Teil von Verfahren des überwachten Lernens analysiert und erweitert. Eine mögliche Anwendung dieser Verfahren soll an einem Einführungsbeispiel nahegelegt werden. Man stellt sich ein Krankenhaus vor, welches verschiedene Daten von Patienten besitzt. Darin sind diverse Eigenschaften der einzelnen Patienten enthalten, wie beispielsweise das Alter, Geschlecht, Blutdruck etc. Außerdem ist zu allen Patienten bekannt, ob diese an einer bestimmten Krankheit leiden oder gesund sind. Anhand dieser Erfahrungsdaten soll für Krankheitsbilder von neuen Patienten eine mögliche Erkrankung oder Nicht-Erkrankung erkannt werden [11].

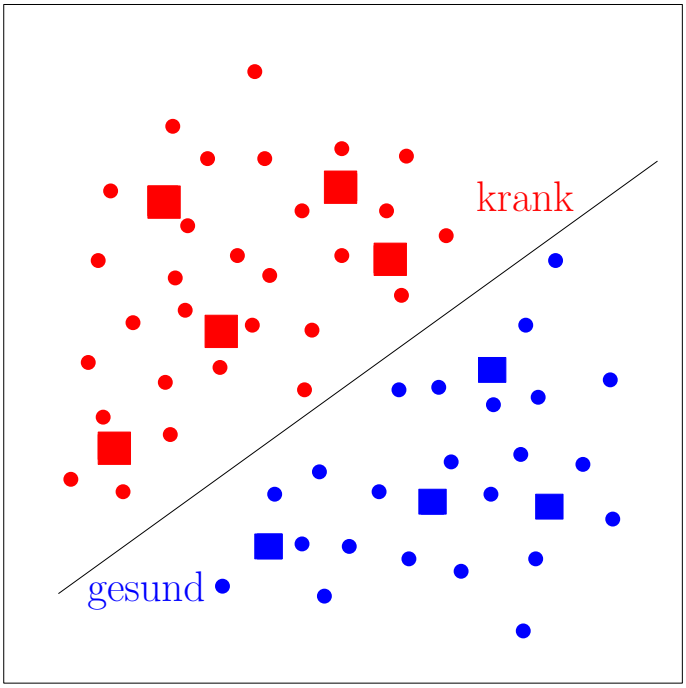

Abbildung 1.1: Prototypen und Datenvektoren für Krankheitsbilder

Demzufolge werden sogenannte Prototypen (Abbildung 1.1: viereckige Punkte) für die Klasse Erkrankung (rot) und Nicht-Erkrankung (blau) bezüglich der Datenvektoren (Abbildung 1.1: runde Punkte) gelernt. Diese Prototypen dienen zur Repräsentation der jeweiligen Klassen (siehe Abbildung 1.1). Die Ähnlichkeit zwischen neuen Krankheitsbildern und der erlernten Prototypen ermöglicht nun eine Zuordnung der Krankheitsbilder zu den jeweiligen Klassen. Dieses Vorgehen ist in der heutigen Medizin durchaus anerkannt und bietet Vorteile in der Diagnose bzw. dient zur Optimierung von Thera-

pieverfahren. In dieser Arbeit werden verschiedene Methoden aus der Familie der LVQ-Verfahren vorgestellt und auf funktionale Daten angepasst. Das bedeutet, man interpretiert die einzelnen Datenvektoren als Folge von Funktionswerten.
In Kapitel 2 werden die Datensätze charakterisiert, welche man für die verschiedenen Tests der Verfahren verwendete. Für die weitere Arbeit führt man in Kapitel 3 grundlegende Begriffe ein. Das erste Verfahren Generalized Learning Vector Quantization, wird im Kapitel 4 vorgestellt. Dieses Verfahren erweitert man mit der vereinfachten Sobolev-Metrik, um zusätzlich die Informationen der Anstiege der Datenvektoren für die Klassifizierung zu berücksichtigen. Außerdem wird eine Möglichkeit aufgezeigt, die Prototypen durch Überlagerungen von Basisfunktionen darzustellen. Hierfür verwendete man die Gaußfunktion und die Sigmoidfunktion als mögliche Basisfunktionen.
Das Kapitel 5 befasst sich mit dem GRLVQ, eine Erweiterung des GLVQ. Der GRLVQ betrachtet zusätzlich relevante Datendimensionen genauer, die für die Klassifikation der Daten wichtig sind. Diesen Algorithmus erweitert man ebenfalls mit der Sobolev-Metrik und der Überlagerung von Basisfunktionen.
Als letztes Verfahren betrachtet man in Kapitel 6 den GMLVQ. Der GMLVQ ist ein sehr mächtiges Werkzeug zur Klassifikation von Datensätzen, vor allem mittels der erlernten Korrelationsmatrix der Datendimensionen. Hierfür werden ebenfalls die Prototypen durch Überlagerungen von Basisfunktionen dargestellt.
Alle betrachteten Verfahren testete man an den Datensätzen, die in Kapitel 2 charakterisiert sind und verarbeitete die jeweiligen Ergebnisse in einer ausführlichen Auswertung. Abschließend wird eine Zusammenfassung dieser Arbeit mit einem zusätzlichen Ausblick auf weitere interessante Aspekte gegeben.

2 Datensätze

Für die Analysen der Verfahren wurden in dieser Arbeit zwei Datensätze aus der Nahrungsmittelbranche gewählt: der Tecator-Datensatz und ein Kaffee-Datensatz. Bei beiden Datensätzen handelt es sich im Spektralmessungen, welche im Folgenden näher charakterisiert werden.

2.1 Tecator-Datensatz

Der Tecator-Datensatz beinhaltet 215 100-dimensionale Spektralmessungen mit einer spektralen Auflösung von 2nm. Aufgenommen wurden diese Messungen vom 'Tecator Infratec Food and Feed Analyzer', der in dem Wellenlängenbereich 850 - 1050nm arbeitet. Hierbei entspricht jede Messung einem Infrarotabsorptionsspektrum einer Fleischprobe. Die Daten sind in zwei Klassen unterteilt, in hohen und niedrigen Fettgehalt. Hoher Fettgehalt bedeutet, die Fleischprobe enthält mehr als 20 % Fett und niedriger Fettgehalt bedeutet weniger als 20% Fett. Der Tecator-Datensatz ist in Abbildung 2.1 illustriert.

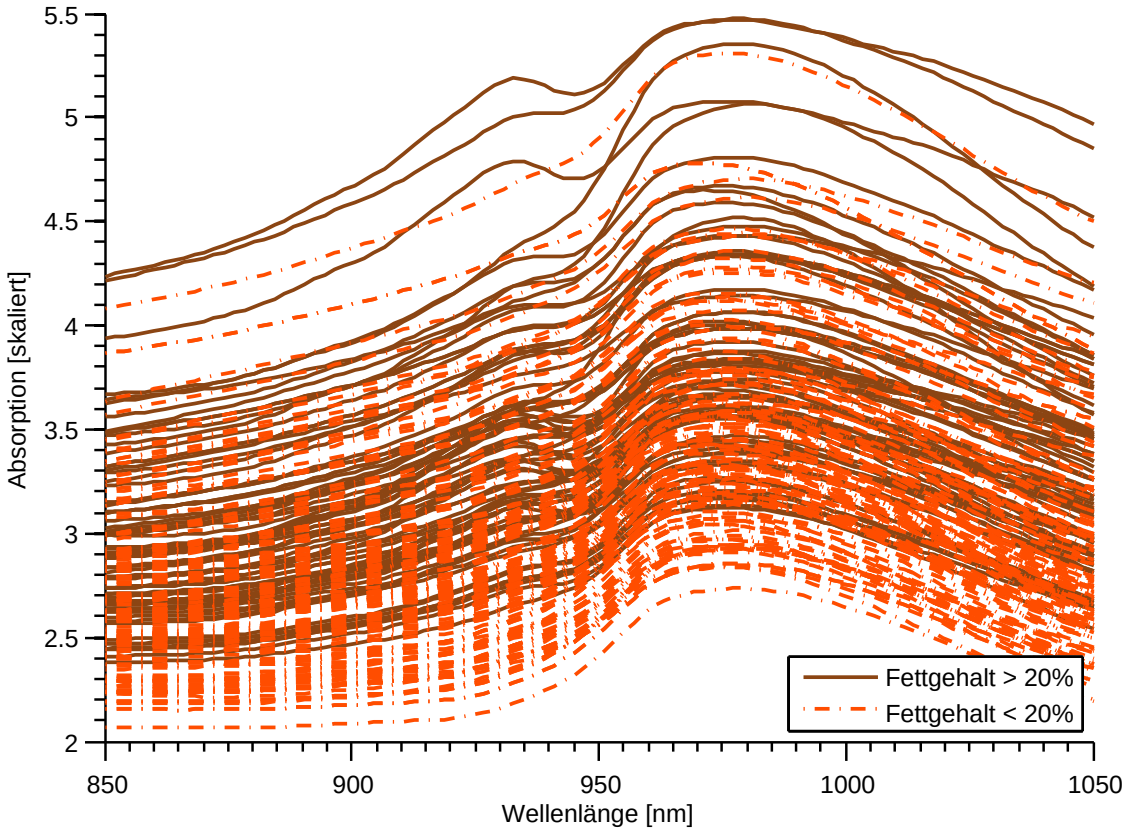

Abbildung 2.1: Tecator-Datensatz

In Abbildung 2.1 ist zu erkennen, dass sich beide Klassen überlagern und im Wesentlichen die Daten die gleiche Form haben. Dies macht eine Klassifikation der Daten schwierig. Dennoch wird auf einen kleinen Unterschied zwischen den beiden Klassen aufmerksam gemacht. Die Daten der Klasse 'hoher Fettgehalt' weisen im Allgemeinen in dem Wellenlängenbereich 920-950nm ein lokales Maximum auf, wobei die andere Klasse 'niedriger Fettgehalt' dieses Merkmal nicht besitzt. Diese Information wird später für eine bessere Klassifizierung der Daten genutzt. Für die jeweiligen Tests der Verfahren wurde der Tecator-Datensatz aufgeteilt in 44% Testdatensatz und 56% Trainingsdatensatz.

2.2 Kaffee-Datensatz

Der Kaffee-Datensatz beinhaltet 2100 256-dimensionale Spektraldaten mit einer spektralen Auflösung von 2nm. Aufgenommen wurden diese Messungen mit der Spektralkamera 'HySpex SWIR-320 m-e' von Norsk Elektro Optikk A/S an dem Fraunhofer-Institut für Fabrikbetrieb und -automatisierung in der Abteilung Biosystems and Engineering Group[1]. Diese arbeitet im Infrarotbereich mit den Wellenlängen 970 - 2500nm. In dem Datensatz aus Abbildung 2.2 entspricht jedes Infrarotabsorptionsspektrum einer bestimmten Kaffeeprobe. Man betrachtete dabei Proben von insgesamt 21 verschiedene Kaffeesorten.

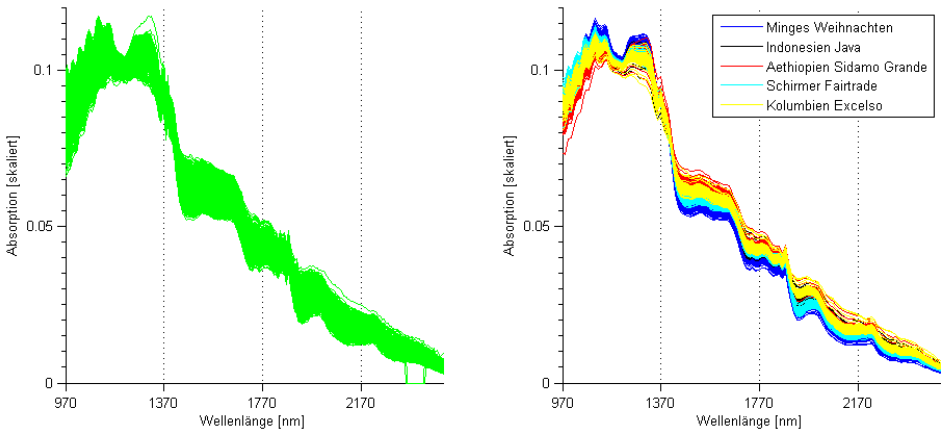

Abbildung 2.2: links: original Kaffee-Datensatz; rechts: reduzierter Kaffee-Datensatz

Für die Tests in dieser Arbeit wurde der Kaffee-Datensatz aus Abbildung 2.2 auf der linken Seite verkleinert, in dem man 5 Kaffeesorten auswählte und eine spektrale Auflösung von 4nm verwendete. Der daraus resultierende Kaffee-Datensatz ist in Abbildung 2.2 auf der rechten Seite illustriert. Für den weiteren Verlauf wird der große Kaffee-Datensatz (linke Seite) als Kaffee-Datensatz I und der reduzierte Kaffee-Datensatz

[1] Ein großer Dank an Udo Seiffert für die Aufnahme und Bereitstellung des Kaffee-Datensatz

(rechte Seite) als Kaffee-Datensatz II bezeichnet. Insgesamt beinhaltet der Kaffee-Datensatz II Messungen von 500 128-dimensionalen Kaffeeproben. Dabei sind die folgenden Kaffeesorten enthalten:

- Minges Weihnachten
- Indonesien Java
- Aethiopien Sidamo Grande
- Schirmer Fairtrade
- Kolumbien Excelso

An dem Kaffee-Datensatz II aus Abbildung 2.2 ist zu erkennen, dass sich die Daten nur teilweise überlagern, aber im Wesentlichen die gleiche Form aufweisen. Außergewöhnliche Unterschiede zwischen den Klassen sind jedoch nicht zu erkennen. Der Kaffee-Datensatz II wurde für die Tests der jeweiligen Verfahren in 20% Test- und 80% Trainingsdatensatz aufgeteilt.

3 Grundbegriffe und Vorbetrachtungen

In diesem Kapitel werden grundlegende Begriffe eingeführt und Vorbetrachtungen angestellt, die für ein besseres Verständnis der Arbeit dienen. Dabei beziehe ich mich im Wesentlichen auf die Literatur [11].
Seien nun $\mathbf{x} = (x(1), \ldots, x(n))^T$ und $\mathbf{y} = (y(1), \ldots, y(n))^T$ Datenvektoren aus einer Datenmenge $\mathscr{X}$ und $d(\mathbf{x}, \mathbf{y})$ ist eine auf dieser Menge $\mathscr{X}$ definierte Metrik. Diese Menge mit der Metrik $d(\mathbf{x}, \mathbf{y})$ bilden einen metrischen Raum, den sogenannten **Datenraum** $\mathscr{X}$. Jedes Elemente des Datenraumes gehört einer **Klassen** $k \in \mathscr{K}$ an, wobei $\mathscr{K}$ die **Klassenmenge** ist. Die eindeutige Zuordnung der Klassen zu den Datenvektoren erfolgt durch die **Klassifikatorfunktion**

$$\kappa : \mathscr{X} \to \mathscr{K} : \kappa(\mathbf{x}) \mapsto k.$$

Somit kann der Datenraum in disjunkte Klassen

$$\mathscr{X}_k = \{\mathbf{x} \in \mathscr{X} \mid \kappa(\mathbf{x}) = k\} \subseteq \mathscr{X}, \forall k \in \mathscr{K}$$

zerlegt werden, so dass

$$\mathscr{X} = \dot{\bigcup_{k \in \mathscr{K}}} \mathscr{X}_k$$

gilt, wobei $\dot{\bigcup}$ für die disjunkte Vereinigung steht. Diese Zerlegung des Datenraumes wird nun als **Klassentrennung** bzw. **Klassenzerlegung** bezeichnet. Mit anderen Worten bildet diese Klassenzerlegung eine Partition des Datenraumes bezüglich der Klassen. Als **Trainingsmenge** $\mathscr{T} \subset \mathscr{X}$ bezeichnet man eine Teilmenge des Datenraumes für die die Klassifikatorfunktion κ bekannt ist.
Die Aufgabe besteht nun darin, anhand der Trainingsdatenmenge $\mathscr{T}$ eine Klassenzerlegung für den gesamten Datenraum $\mathscr{X}$ zu lernen. Es soll somit die gegebene Klassifikatorfunktion κ für $\mathscr{T}$ erlernt und auf den gesamten Datenraum angewendet werden. Die erlernte Funktion wird mit κ' bezeichnet.

Bemerkung 3.1 Es wird noch einmal darauf hingewiesen, dass κ die bereits bekannte theoretische Klassifikatorfunktion der Trainingsdatenmenge ist und κ' die durch ein Verfahren erlernte Klassifikatorfunktion. Das Ziel ist nun k' so zu lernen, dass $k'(\mathbf{x}) = k(\mathbf{x})$, $\forall \mathbf{x} \in \mathscr{T}$.

Damit die Abbildung κ nicht auswendig gelernt wird, wird der Datenraum $\mathscr{X}$ aufgeteilt in den Trainingsraum $\mathscr{T}$ und in den Testdatenraum $\mathscr{T}' = \mathscr{X} \setminus \mathscr{T}$. Da für den Trainingsraum $\mathscr{T}$ die Klassifikatorfunktion κ gegeben ist, kann die erlernte Klassifikatorfunktion κ' stets mit κ auf ihre Güte verglichen werden. Ist eine relative hohe Güte gewährleistet für den Datensatz, so kann die Funktion κ' auf den Testdatensatz übertragen werden. Die Güte der einzelnen Verfahren wird durch die **Genauigkeitsrate** r_a angegeben. Diese

lässt sich mit der Formel

$$r_a = \frac{1}{\mid \mathscr{T} \mid} \sum_{\mathbf{x} \in \mathscr{T}} g(\mathbf{x}),$$

berechnen, wobei

$$g(\mathbf{x}) = \begin{cases} 1, & \kappa(\mathbf{x}) = \kappa'(\mathbf{x}) \\ 0, & \kappa(\mathbf{x}) \neq \kappa'(\mathbf{x}) \end{cases}$$

In dieser Arbeit werden ausschließlich prototypenbasierende Verfahren des überwachten Lernens betrachtet. Hierbei benötigt man für die Lernaufgabe Repräsentanten der einzelnen Klassen $k \in \mathscr{K}$, sogenannte **Prototypen** $\mathbf{w}_i^k$. Die Menge aller Prototypen der Klasse $k \in \mathscr{K}$ wird mit $W^k = \{\mathbf{w}_i^k \mid i = 1, ..., n_k\}$ bezeichnet, wobei $W = \dot{\bigcup_k} W^k$ die Menge aller Prototypen ist. Die Prototypen werden nun so in dem Datenraum $\mathscr{X}$ platziert, dass sie möglichst gut die Informationen ihrer jeweiligen Klasse wiedergeben. Dabei ist der Effekt des 'auswendig Lernens' unbedingt zu vermieden. Das bedeutet, dass zwar der Trainingsfehler für die Trainingsmenge sehr klein ist, jedoch der Generalisierungsfehler für die Testdaten mit den gelernten Prototypen sehr hoch. Um dieses Problem zu umgehen, ist die Anzahl der Prototypen für die einzelnen Klassen günstig zu wählen und deren Lage im Datenraum optimal zu bestimmen. Im weiteren Verlauf wird diese optimale Lage noch näher erklärt. Die dabei erlernte Klassenaufteilung von $\mathscr{T}$ kann stets mit der vorhandenen Klassifikatorfunktion nachgeprüft werden.
Es wird also den Datenvektoren aus $\mathscr{X}$ eine Klasse anhand der Prototypen zugeordnet. Diese Klassifikation der Datenvektoren basiert auf dem **Prinzip des kleinsten Abstandes**

$$\mathbf{x} \in \mathscr{X}_k \;\Leftrightarrow\; \min_i d(\mathbf{x}, \mathbf{w}_i^k) \leq \min_j d(\mathbf{x}, \mathbf{w}_j^l),\ \forall l \neq k. \tag{3.1}$$

Jedem Datenpunkt wird die Klasse zugeordnet, deren Prototyp am nächsten zu diesem Datenpunkt liegt. Sollten zwei Prototypen mit dem gleichen Abstand am nächsten an dem Datenvektor $\mathbf{x}$ liegen, wird durch Zufall einer von beiden ausgewählt, nach dem gelabelt wird.

4 Generalized Learning Vector Quantization

4.1 Theorie Generalized Learning Vector Quantization

Dieser Abschnitt basiert vorwiegend auf der Literatur [14], [17] und [11].
'Generalized Learning Vector Quantization' (GLVQ) wurde 1993 von A. Sato und K. Yamada veröffentlicht. Die Idee besteht darin, 'Learning Vector Quantization (LVQ)' von Kohonen [15] zu verallgemeinern. Sei nun $\mathscr{T} = \{(\mathbf{x},k) \mid \mathbf{x} \in \mathbb{R}^n,\ k \in \mathscr{K}\}$ eine Trainingsmenge mit n-dimensionalen Datenvektoren und $|\mathscr{K}|$ Klassen. Nach dem Grundprinzip des LVQ von Kohonen wählt man sich in jedem Lernschritt einen zufälligen Datenvektor $\mathbf{x} = (x(1),...,x(n)) \in \mathscr{T}$ mit der Klasse $\kappa(\mathbf{x})$ aus. Anhand dieses Datenvektors werden die zwei nächsten Prototypen ermittelt. Oft wird hierfür das quadratisch euklidische Ähnlichkeitsmaß

$$d(\mathbf{x},\mathbf{w}) := (\|\mathbf{x}-\mathbf{w}\|_2)^2 = \sum_{t=1}^{n} (x(t)-w(t))^2.$$

verwendet.

Bemerkung 4.1 Es wird hierbei angemerkt, dass das quadratisch euklidische Ähnlichkeitsmaß eine Quasimetrik ist. Eine Quasimetrik erfüllt die gleichen Eigenschaften wie eine Metrik, bis auf die Dreiecksungleichung. Für nähere Ausführungen wird auf die Literatur [8] verwiesen. Dennoch wird im weiteren Verlauf von einer Metrik gesprochen.

Diese Prototypen werden nun wie folgt adaptiert:

- Sind nun beide Prototypen von derselben Klasse wie der Datenvektor, werden die beiden Prototypen nicht adaptiert.
- Ist nur ein Prototyp von derselben Klasse, wird dieser zu dem Datenvektor hingezogen und der andere Prototyp von einer anderen Klasse wird weggestoßen.
- Sind beide Prototypen von einer anderen Klasse, werden beide Prototypen von dem Datenvektor weggestoßen.

Es wird an dieser Stelle darauf hingewiesen, dass es sich bei den Update-Regeln um den LVQ 2.1 handelt. In [15] von Kohonen werden weitere Varianten des LVQ angegeben. Der LVQ von Kohonen löst das Klassifikationsproblem für viele Fälle und erzielt respektable Ergebnisse. Jedoch zeigten Sato und Yamada in [14], dass der LVQ von Kohonen im Allgemeinen nicht in einem globalen Optimum konvergiert.
Aus diesem Grund führten Sato und Yamada eine Kostenfunktion ein, welche die Anzahl der fehlklassifizierten Datenvektoren approximiert. Diese Kostenfunktion kann durch einen stochastischen Gradientenabstieg minimiert werden. Dadurch werden die Prototypen während des Lernprozesses sukzessive adaptiert, so dass eine möglichst fehlerfreie Klassenzerlegung des Datenraumes approximiert wird.

Ähnlich wie bei dem LVQ wird bei dem GLVQ in Datenvektor $\mathbf{x}$ zufällig ausgewählt. Des Weiteren wird nun der Gewinnerprototyp $\mathbf{w}^+$ ermittelt, der am nächsten an dem ausgewählten Datenvektor $\mathbf{x} \in \mathscr{T}$ liegt und von der selben Klasse ist wie $\mathbf{x}$, d.h. $\kappa(\mathbf{w}^+) = \kappa(\mathbf{x})$. Außerdem ermittelt man den Verliererprototyp $\mathbf{w}^-$, der am nächsten zu dem Datenvektor $\mathbf{x} \in \mathscr{T}$ liegt und von einer anderen Klasse ist, d.h. $\kappa(\mathbf{w}^+) \neq \kappa(x)$. Der Gewinnerprototyp $\mathbf{w}^+$ wird nun zu dem Datenvektor $\mathbf{x}$ hingezogen und $\mathbf{w}^-$ wird von $\mathbf{x}$ weggestoßen. Hierfür betrachtet man zunächst die Funktion des relativen Fehlers

$$\mu(\mathbf{x}) = \frac{d^+(\mathbf{x}) - d^-(\mathbf{x})}{d^+(\mathbf{x}) + d^-(\mathbf{x})}, \tag{4.1}$$

wobei $d^+(\mathbf{x}) := d(\mathbf{x}, \mathbf{w}^+)$ der Gewinnerabstand und $d^-(\mathbf{x}) := d(\mathbf{x}, \mathbf{w}^-)$ der Verliererabstand ist. Diese Funktion besitzt einen Wertebereich im Intervall $[-1, 1]$. Es kann leicht nachgeprüft werden, dass bei einer falschen Klassifizierung des Datenvektors $\mathbf{x} \in \mathscr{T}$ der Funktionswert $\mu(\mathbf{x})$ positiv und bei einer richtigen Klassifizierung negativ ist. [2] Die Approximation des Klassifikationsfehlers wird somit durch die Kostenfunktion

$$E^{GLVQ} = \sum_{\mathbf{x} \in \mathscr{T}} f(\mu(\mathbf{x})) \tag{4.2}$$

berechnet, wobei f eine monoton wachsende Funktionen ist.

Bemerkung 4.2 Für die späteren Testergebnisse wurde für f stets die Sigmoidfunktion

$$f(\mu(\mathbf{x})) = \frac{1}{1 + e^{-\sigma \cdot \mu(\mathbf{x})}}$$

mit $\sigma = 1$ verwendet.

Die Kostenfunktion E^{GLVQ} wird nun durch einen stochastischen Gradientenabstieg minimiert. Man erhält somit für den Gewinnerprototypen $\mathbf{w}^+$ und den Verliererprototypen $\mathbf{w}^-$ die Adaptionsregel

$$\mathbf{w}^\pm \leftarrow \mathbf{w}^\pm - \varepsilon_r \frac{\partial E^{GLVQ}}{\partial \mathbf{w}^\pm},$$

wobei ε_r die Lernrate im r-ten Trainingsschritt ist. Es wird nochmal darauf hingewiesen, dass es sich bei dem Gradienten um einen stochastischen Gradienten handelt. Unter der Verwendung der quadratisch euklidischen Metrik ergeben sich für die Kostenfunktion (4.2) die folgenden Gradienten:

$$\frac{\partial E^{GLVQ}}{\partial \mathbf{w}^+} = \frac{\partial E^{GLVQ}}{\partial \mu} \cdot \frac{\partial \mu}{\partial d^+(\mathbf{x})} \cdot \frac{\partial d^+(\mathbf{x})}{\partial \mathbf{w}^+}$$

[2] Bekannt ist diese Funktion auch unter dem Namen 'classifier function'.

$$= -\frac{\partial f}{\partial \mu} \cdot \frac{4d^-(\mathbf{x})}{(d^+(\mathbf{x})+d^-(\mathbf{x}))^2} \cdot (\mathbf{x}-\mathbf{w}^+) \tag{4.3}$$

$$\frac{\partial E^{GLVQ}}{\partial \mathbf{w}^-} = \frac{\partial E^{GLVQ}}{\partial \mu} \cdot \frac{\partial \mu}{\partial d^-(\mathbf{x})} \cdot \frac{\partial d^-(\mathbf{x})}{\partial \mathbf{w}^-}$$

$$= \frac{\partial f}{\partial \mu} \cdot \frac{4d^+(\mathbf{x})}{(d^+(\mathbf{x})+d^-(\mathbf{x}))^2} \cdot (\mathbf{x}-\mathbf{w}^-) \tag{4.4}$$

Dem zufolge ergeben sich für jeden zufällig gewählten Datenvektor $\mathbf{x} \in \mathscr{T}$ die Adaptionsregeln des GLVQ:

$$\mathbf{w}^+ \leftarrow \mathbf{w}^+ + \varepsilon_r \frac{\partial f}{\partial \mu} \cdot \frac{d^-(\mathbf{x})}{(d^+(\mathbf{x})+d^-(\mathbf{x}))^2} \cdot (\mathbf{x}-\mathbf{w}^+) \tag{4.5}$$

$$\mathbf{w}^- \leftarrow \mathbf{w}^- - \varepsilon_r \frac{\partial f}{\partial \mu} \cdot \frac{d^+(\mathbf{x})}{(d^+(\mathbf{x})+d^-(\mathbf{x}))^2} \cdot (\mathbf{x}-\mathbf{w}^-) \tag{4.6}$$

Der Gewinnerprototyp $\mathbf{w}^+$ wird somit in Richtung des Datenvektors $\mathbf{x} \in \mathscr{T}$ geschoben und $\mathbf{w}^-$ wird weggestoßen.
Kushner und Clark zeigten in [7], dass ein solcher Lernprozess für $r \to \infty$ in einem globalen Minimum konvergiert, wenn die Lernrate ε_r die folgenden Bedingungen erfüllt:

$$1.)\ 0 < \varepsilon_r \ll 1 \tag{4.7}$$

$$2.)\ \sum_{r=1}^{\infty} \varepsilon_r = \infty \tag{4.8}$$

$$3.)\ \sum_{r=1}^{\infty} \varepsilon_r^2 < \infty \tag{4.9}$$

Bemerkung 4.3 Eine mögliche Wahl der Lernrate ist $\varepsilon_r = \frac{1}{r}$, welche die obigen drei Bedingungen erfüllt. Mit anderen Worten muss die Lernrate ε_r zur Konvergenz des Verfahrens mit fortschreitender Zeit abkühlen.

Dabei wird nochmals verdeutlicht, dass $r \to \infty$ in der Praxis nicht realisierbar ist. Bricht man den Lernprozess also nach endlich vielen Schritten ab, so kann möglicherweise nur ein lokales Minimum der Kostenfunktion erreicht worden sein.
Wiederholt man also den stochastischen Gradientenabstieg für zufällig gewählte Datenvektoren, nähern sich die Prototypen einer Lage im Datenraum an, so dass der Funktionswert der Kostenfunktion (4.2) minimiert wird. In dem folgenden Algorithmus sind die wesentlichsten Schritte des GLVQ zusammengefasst.

Algorithm 1 GLVQ

Eingabe: Trainingsdatensatz $\mathscr{T}$, maximale Anzahl der Iterationen r_{max}, Anzahl der Prototypen pro Klasse und Lernrate ε
Ausgabe: Prototypen
zufällige Initialisierung der Prototypen $\mathbf{w}$ für jede Klasse
Setze r:=0

while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Berechne alle Abstände der Prototypen $\mathbf{w}$ zu dem Datenpunkt $\mathbf{x}$
  Ermittle den zu $\mathbf{x} \in \mathscr{T}$ gehörigen Gewinner- $\mathbf{w}^+$ und Verliererprototyp $\mathbf{w}^-$
  Adaptiere Gewinner- und Verliererprototyp nach Gleichungen (4.5) und (4.6)
  $r = r + 1$
end while

4.2 Modifikation des GLVQ

In diesem Abschnitt werden verschiedene Ansätze vorgestellt, die eine mögliche Verbesserung des GLVQ hinsichtlich der Genauigkeitsrate und der Generalisierung bewirken. Zunächst wird der GLVQ dahingehend modifiziert, dass als Metrik die Sobolev-Metrik eingeführt wird. Im weiteren Verlauf werden zusätzlich die Prototypen als Überlagerungen von Basisfunktionen dargestellt.

4.2.1 GLVQ mit Sobolev-Norm

In Abschnitt 4.1 wurde der GLVQ von Sato und Yamada vorgestellt. In dem GLVQ wird die euklidische Metrik für die Berechnung der Abstände zwischen Datenvektoren $\mathbf{x}$ aus der in Abschnitt 4.1 definierten Trainingsmenge $\mathscr{T}$ und den Prototypen $\mathbf{w}$ verwendet. Sei nun $\mathscr{T} = \{(\mathbf{x},k) \mid \mathbf{x} \in \mathbb{R}^n,\ k \in \mathscr{K}\}$ eine Trainingsmenge, deren Daten $\mathbf{x}$ einen funktionalen Zusammenhang aufweisen. Das bedeutet, die Komponenten der Datenvektoren $\mathbf{x} = (x(1),\ldots,x(n))^T$ werden als diskrete Repräsentanten von stetig-differenzierbaren Funktionen interpretiert. Eine Darstellung von funktionalen Daten ist in Abbildung 2.1 zu erkennen. Aus diesem Grund wird die Annahme getroffen, dass die Daten im $C^1[1,n]$ liegen, dem Raum der einmal stetig-differenzierbaren Funktionen auf dem Intervall $[1,n]$.

Bemerkung 4.4 Die gegebenen Daten liegen als Vektoren vor. Die Komponenten eines einzelnen Vektors werden als Folge von Funktionswerten interpretiert in Abhängigkeit von $t \in \{1,\ldots,n\}$. Des Weiteren wird jedoch nicht von Funktionen gesprochen sondern von Datenvektoren.

Es wird nun eine Variante des GLVQ vorgestellt, der diese funktionale Eigenschaft der Datenvektoren zur Klassifizierung nutzt und zusätzlich Informationen über die Anstiege der Daten verarbeitet. Zu diesem Zweck definiert man für $1 \leq p < \infty$ die Sobolev-Norm

$$\|\mathbf{x}\|_{S(p,R)} := \left(\sum_{j=0}^{R} \|D^{(j)}\mathbf{x}\|_{\mathscr{L}^p}^p \right)^{\frac{1}{p}}, \tag{4.10}$$

wobei $\|\cdot\|_{\mathscr{L}^p}$ die $\mathscr{L}^p$-Norm und $D^{(j)}$ der Differenzialoperator der j-ten Ordnung ist.

Weiter sei R die höchste Ordnung der Ableitung, die in die Berechnung der Sobolev-Norm eingeht. Außerdem ist $D^{(j)}\mathbf{x} := \frac{\partial^j x(t)}{\partial t^j}$ die j-te partielle Ableitung von $\mathbf{x}$ nach t. Der dazugehörige diskrete Repräsentant der Ableitung wird mit $x^{(j)}(t)$ bezeichnet. Analog gilt das Gleiche für die Prototypen $\mathbf{w}$.
Es kann gezeigt werden[3], dass (4.10) äquivalent ist zu der Norm

$$\|\mathbf{x}\|_{p,R} := \left(\|\mathbf{x}\|^p_{\mathscr{L}^p} + \|D^{(R)}\mathbf{x}\|^p_{\mathscr{L}^p}\right)^{\frac{1}{p}}. \tag{4.11}$$

Des Weiteren wird der Ausdruck (4.11) als vereinfachte Sobolev-Norm bezeichnet. Im Folgenden betrachtet man den Fall $R = 1$ und $p = 2$ für (4.11). Damit lässt man in die vereinfachte Sobolev-Norm lediglich die Original-Funktion und deren erste Ableitung einfließen. Mit dieser Darstellung werden somit zusätzlich Informationen über die Anstiege der Datenvektoren genutzt. Es ergibt sich somit folgender Ausdruck

$$\|\mathbf{x}\|_{2,1} = \left(\|\mathbf{x}\|^2_E + \|\dot{\mathbf{x}}\|^2_E\right)^{\frac{1}{2}}, \tag{4.12}$$

wobei $\|\cdot\|_E$ die euklidische Norm ist und $\dot{\mathbf{x}} = D^{(1)}\mathbf{x} = \frac{\partial x(t)}{\partial t}$.

Bemerkung 4.5 Es wird in dem Ausdruck (4.12) die euklidische Norm für den diskreten Fall verwendet, da es sich bei den Daten um Vektoren handelt. Das Analogon für den stetigen Fall von (4.12) ist

$$\begin{aligned}\|\mathbf{x}\|_{2,1} &= \left(\|\mathbf{x}\|^2_{L^2(\Omega)} + \|\dot{\mathbf{x}}\|^2_{L^2(\Omega)}\right)^{\frac{1}{2}} \\ &= \left(\int_0^1 |\, x(t)\,|^2\, dt + \int_0^1 |\, \dot{x}(t)\,|^2\, dt\right)^{\frac{1}{2}}\end{aligned}$$

Wenn keine Ableitungen der Datenvektoren bzw. Prototypen gegeben sind, müssen diese approximativ berechnet werden. Die Sobolev-Metrik gewährleistet eine Regularisierung der Prototypen in ihrer Darstellung [19].
Für die Glättung der Datenvektoren wurden in der vorliegenden Arbeit die Methode der gleitenden Mittel angewendet. Das hat den Vorteil, dass das möglicherweise vorhandenes Rauschen in den einzelnen Dimensionen reduziert wird und somit Fehler vermieden werden können. Es wird dabei eine ungeradzahliger Ordnung angewendet, damit sowohl die linke als auch die rechte Seite gleichmäßig für die Glättung verwendet werden. Jeder Datenvektor $\mathbf{x} := (x(1),\ldots,x(n))^T$ wurde beispielsweise mit der Ordnung 3,

[3] Ein Beweis für die Äquivalenz befindet sich im Anhang.

wie folgt geglättet:

$$\begin{aligned}\bar{x}(1) &= x(1)\\ \bar{x}(l) &= \frac{x(l-1)+x(l)+x(l+1)}{3},\ \forall l \in [2,n-1]\\ \bar{x}(n) &= x(n)\end{aligned}$$

Dabei ist $\bar{\mathbf{x}} := (\bar{x}(1),\ldots,\bar{x}(n))^T$ der geglättete Datenvektor. Je größer die Ordnung ist, umso weniger geglättete Randwerte als Komponenten hat man nach der Glättung zur Verfügung. In der Praxis wurde demnach für die geglätteten Randwerte stets die ungeglätteten Rohdaten verwendet. Aus den geglätteten Datenvektoren errechnete man die numerischen Ableitungen über den zentralen Differenzenquotienten [10]. Die Ableitungen des Datenvektors $\mathbf{x}$ wurden demnach mit

$$\dot{x}(t) \approx \frac{\Delta x}{\Delta t} = \frac{x(t+1)-x(t-1)}{2},\ \forall t \in [2,n-1]$$

ermittelt. Für die Randwerte $\dot{x}(1)$ bzw. $\dot{x}(n)$ verwendete man den Vorwärts- bzw. Rückwärtsdifferenzenquotienten mit

$$\begin{aligned}\dot{x}(1) &\approx x(2)-x(1)\\ \dot{x}(n) &\approx x(n)-x(n-1).\end{aligned}$$

Die vereinfachte Sobolev-Norm (4.12) induziert nun eine Metrik, die sogenannte Sobolev-Metrik. Mit dieser werden die Abstände zwischen den Prototypen $\mathbf{w}$ und den Datenvektoren $\mathbf{x}$ berechnet. Um später zusätzlich Aussagen bzgl. der Relevanz der ersten Ableitung bei der Klassifikation der Datenvektoren machen zu können, wichtet man den hinteren Term mit dem Parameter $\alpha \in [0,1]$ und den vorderen Teil mit $(1-\alpha)$. Es ergibt sich für die Sobolev-Metrik die folgende Formel:

$$\begin{aligned}d_S(\mathbf{x},\mathbf{w},\alpha) &:= (\|\mathbf{x}-\mathbf{w}\|_{2,1})^2\\ &= (1-\alpha)\cdot(\|\mathbf{x}-\mathbf{w}\|_E)^2+\alpha(\|\dot{\mathbf{x}}-\dot{\mathbf{w}}\|_E)^2\\ &= (1-\alpha)\cdot\sum_{t=1}^{n}(x(t)-w(t))^2+\alpha\cdot\sum_{t=1}^{n}(\dot{x}(t)-\dot{w}(t))^2. \qquad (4.13)\end{aligned}$$

Bemerkung 4.6 Es gilt hier die gleiche Vereinbarung, wie bei der quadratisch euklidischen Metrik: Obwohl $d_S(\mathbf{x},\mathbf{w},\alpha)$ eine Quasimetrik ist [8], wird weiterhin von der quadratischen Sobolev-Metrik gesprochen.

Somit werden für die Berechnung der Distanzen im ersten Term der Sobolev-Metrik die originalen Datenvektoren $\mathbf{x}$ genommen und für die Distanzen des zweiten Terms die Ableitungen der geglätteten Datenvektoren $\bar{\mathbf{x}}$. Analog gilt das gleiche für die Prototypen. Nun wird deutlich, wieso lediglich die erste Ableitung der Datenvektoren bzw. Prototypen in die Sobolev-Metrik eingeht. Durch die Approximation der Ableitungen der Datenvekto-

ren bzw. Prototypen entsteht ein gewisser Fehler. Dieser würde sich bei der Berechnung von Ableitungen höherer Ordnung verstärken. Aus diesem Grund betrachtet man oft nur die erste Ableitung, um die Information über die Anstiege zu berücksichtigen und den dabei entstehenden Fehler klein zu halten.
Im Folgenden soll neben der Adaption der Prototypen auch der Wichtungsparameter α durch einen stochastischen Gradientenabstieg optimiert werden. Der Parameter α kann als Relevanzfaktor der ersten Ableitungen bei der Erstellung der Klassenzerlegung interpretiert werden. Das bedeutet, er gibt den notwendigen Anteil der Ableitungen zur Klassifizierung der Daten an. Die Modifikation des GLVQ soll neben der Adaption der Prototypen nun auch α adaptieren, um gegebenenfalls eine optimale Belegung von α zur Minimierung der neuen Kostenfunktion (bzgl. der Sobolev-Metrik)

$$E_S^{GLVQ} = \sum_{\mathbf{x}\in\mathscr{T}} f(\mu_S(\mathbf{x})) \tag{4.14}$$

angeben zu können, wobei

$$\mu_S(\mathbf{x}) = \frac{d_S^+(\mathbf{x}) - d_S^-(\mathbf{x})}{d_S^+(\mathbf{x}) + d_S^-(\mathbf{x})}.$$

Für die Herleitung der Update-Regeln werden die Größen $d_S^{\pm}(\mathbf{x})$ genauso interpretiert wie $d^{\pm}(\mathbf{x})$ aus Abschnitt 4.1. Es wird lediglich die Sobolev-Metrik verwendet. Wie schon bei dem klassischen GLVQ, basiert die Adaption der Prototypen auf einem stochastischen Gradientenabstieg

$$\mathbf{w}^{\pm} \leftarrow \mathbf{w}^{\pm} - \varepsilon_r \frac{\partial E_S^{GLVQ}}{\partial \mathbf{w}^{\pm}},$$

wobei ε_r die Lernrate des r-ten Lernschrittes ist. Ed werden zunächst die drei Gradienten der Kostenfunktion (4.14) nach $\mathbf{w}^+$, $\mathbf{w}^-$ und zusätzlich den Wichtungsparameter α angegeben:

$$\begin{aligned}
\frac{\partial E_S^{GLVQ}}{\partial \mathbf{w}^+} &= \frac{\partial E_S^{GLVQ}}{\partial \mu_S} \cdot \frac{\partial \mu_S}{\partial d_S^+(\mathbf{x})} \cdot \frac{\partial d_S^+(\mathbf{x})}{\partial \mathbf{w}^+} \\
&= \frac{\partial f}{\partial \mu_S} \cdot \frac{2 \cdot d_S^-(\mathbf{x})}{(d_S^+(\mathbf{x}) + d_S^-(\mathbf{x}))^2} \cdot \frac{\partial d_S^+(\mathbf{x})}{\partial \mathbf{w}^+} \\
\frac{\partial E_S^{GLVQ}}{\partial \mathbf{w}^-} &= \frac{\partial E_S^{GLVQ}}{\partial \mu_S} \cdot \frac{\partial \mu_S}{\partial d_S^-(\mathbf{x})} \cdot \frac{\partial d_S^-(\mathbf{x})}{\partial \mathbf{w}^-} \\
&= -\frac{\partial f}{\partial \mu_S} \cdot \frac{2 \cdot d_S^+(\mathbf{x})}{(d_S^+(\mathbf{x}) + d_S^-(\mathbf{x}))^2} \cdot \frac{\partial d_S^-(\mathbf{x})}{\partial \mathbf{w}^-} \\
\frac{\partial E_S^{GLVQ}}{\partial \alpha} &= \frac{\partial f}{\partial \mu_S} \cdot \frac{2\left(\frac{\partial d_S^+(\mathbf{x})}{\partial \alpha} \cdot d_S^-(\mathbf{x}) - \frac{\partial d_S^-(\mathbf{x})}{\partial \alpha} \cdot d_S^+(\mathbf{x})\right)}{(d_S^+(\mathbf{x}) + d_S^-(\mathbf{x}))^2},
\end{aligned}$$

wobei

$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \mathbf{w}^{\pm}} = -(1-\alpha)\cdot(\mathbf{x}-\mathbf{w}^{\pm})$$
$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \alpha} = (\|\dot{\mathbf{x}}-\dot{\mathbf{w}}^{\pm}\|_E)^2 - (\|\mathbf{x}-\mathbf{w}^{\pm}\|_E)^2.$$

Zusammenfassend ergeben sich für den GLVQ mit der Sobolev-Metrik die folgenden Update-Regeln für die Prototypen und den Wichtungsparameter:

$$\mathbf{w}^+ \leftarrow \mathbf{w}^+ + \varepsilon_r \frac{\partial f}{\partial \mu_S}\cdot\frac{d_S^-(\mathbf{x})\cdot(1-\alpha)\cdot(\mathbf{x}-\mathbf{w}^+)}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2} \tag{4.15}$$

$$\mathbf{w}^- \leftarrow \mathbf{w}^- - \varepsilon_r \frac{\partial f}{\partial \mu_S}\cdot\frac{d_S^+(\mathbf{x})\cdot(1-\alpha)\cdot(\mathbf{x}-\mathbf{w}^-)}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2} \tag{4.16}$$

$$\alpha \leftarrow \alpha - \varepsilon_r^{\alpha}\frac{\partial f}{\partial \mu_S}\cdot\frac{\left(\frac{\partial d_S^+(\mathbf{x})}{\partial \alpha}\cdot d_S^-(\mathbf{x}) - \frac{\partial d_S^-(\mathbf{x})}{\partial \alpha}\cdot d_S^+(\mathbf{x})\right)}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2} \tag{4.17}$$

Des Weiteren gilt für die Lernrate des Parameters α:

$$\varepsilon_r^{\alpha} << \varepsilon_r \tag{4.18}$$

Bemerkung 4.7 Diese Lernrate ε^{α} wird deutlich kleiner gewählt als die Lernrate ε der Prototypen. Bei dem endliche Lernen erzwingt man Stabilität, indem man der Konvergenz der Prototypen den Vorrang gibt. Das bedeutet, die Metrik wird quasi-stationär angenommen und annähernd adiabatisch verändert. [4] Somit schenkt man der Konvergenz der Prototypen in dieser Arbeit eine höhere Priorität als die Konvergenz der Metrik.

Die wesentlichsten Schritte des GLVQ mit der Sobolev-Metrik sind im Algorithmus 2 nochmals zusammengefasst.

Algorithm 2 GLVQ mit Sobolev-Norm

Eingabe: Lernraten ε und ε^{α}, Trainingsdatensatz $\mathscr{T}$, Startwert α, maximale Anzahl der Iterationen r_{max} und Anzahl der Prototypen pro Klasse
Ausgabe: Prototypen
Ermittle geglättete Datenpunkte $\overline{\mathbf{x}}$
Ermittle erste Ableitung der geglätteten Datenpunkte $\dot{\overline{\mathbf{x}}}$
zufällige Initialisierung der Prototypen $\mathbf{w}$
Setze r:=0
while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Ermittle geglättete Prototypen $\overline{\mathbf{w}}$ und deren Ableitung $\dot{\overline{\mathbf{w}}}$ approximiert

[4] Die Bezeichnung des 'adiabatischen Abkühlens bzw. der adiabatischen Zustandsänderung' ist eine Beobachtung aus der Physik und wurde das erste Mal von Kästner, Biehl, Villmann und Hammer in [4] eingeführt.

Ermittle euklidischen Abstand zwischen allen Prototypen $\mathbf{w}$ und dem Datenpunkt $\mathbf{x}$
Ermittle euklidischen Abstand zwischen allen geglätteten Prototypen $\dot{\overline{\mathbf{w}}}$ und dem geglätteten Datenpunkt $\dot{\overline{\mathbf{x}}}$
Berechne Abstände aller Prototypen zum Datenpunkt $\mathbf{x}$ mit (4.13)
Ermittle Gewinner- und Verliererprototyp von Datenpunkt $\mathbf{x}$
Adaptiere Gewinner- und Verliererprototyp nach Gleichungen (4.15) und (4.16)
Adaptiere den Parameter α mit Gleichung (4.17)
$r = r + 1$
end while

4.2.2 Prototypen als Überlagerung von Basisfunktionen

Bisher adaptierte man die Prototypen durch einen stochastischen Gradientenabstieg, indem jede einzelne Komponente des Vektors gelernt wurde. In diesem Abschnitt wird die funktionale Eigenschaft der gegebenen Datenvektoren dahingehend genutzt, dass die Prototypen als Überlagerung von Basisfunktionen dargestellt werden. Im Großen und Ganzen wird somit ein Basissystem mit endlich vielen Basisfunktionen für die Datenvektoren adaptiert. Dies hat den Vorteil, dass die Anzahl der zu lernenden Parameter reduziert wird, indem lediglich die Parameter der einzelnen Basisfunktionen gelernt werden müssen. Außerdem besteht zusätzlich der Vorteil in dem hinteren Term der Sobolev-Metrik mit analytischen Ableitungen der Prototypen rechnen zu können.
Sei nun $\mathscr{B} = \{b_i(t) \mid i \in \mathscr{I}\}$ ein Basissystem des $\mathscr{C}^1[1,n]$ von linear unabhängigen Funktionen, wobei $\mathscr{I}$ eine Indexmenge ist und n die Dimension der Datenvektoren.

Bemerkung 4.8 Im weiteren Verlauf wird deutlich, dass es sich bei der Indexmenge um eine unendliche Menge handelt. Die Linearkombination wird aber notwendigerweise endlich viele Basisfunktionen gebildet, da die Vektoren als Repräsentanten endlichdimensional sind. Demnach gilt für die Praxis $G = \mid \mathscr{B}' \mid < n$, wobei die Menge der verwendeten Basisfunktionen $\mathscr{B}' \subset \mathscr{B}$ ist. Ein weiterer Vorteil wäre dabei die Reduktion der zu erlernenden Parameter. Hierbei gilt: $p \cdot G << n$, wobei p die Anzahl der zu erlernenden Parameter pro Basisfunktion ist.

Die Sobolev-Metrik und Bemerkung 4.8 gewährleisten nun die Eindeutigkeit der Prototypen in ihrer Darstellung [19]. Es lässt sich jedes Element aus dem $\mathscr{C}^1[1,n]$ durch eine Linearkombination der Basisfunktionen aus $\mathscr{B}$ darstellen. In der Praxis ist die Wahl der Basisfunktionen stark von der Form des betrachteten Datensatzes abhängig. Das bedeutet, die Basisfunktionen müssen zur Darstellung der Datenvektoren günstig gewählt werden. Es werden in diesem Kapitel die Ansätze mit Basissystemen von Gaußfunktionen und Sigmoidfunktionen vorgestellt. Villmann und Kästner geben in [16] die Lorentzfunktion als eine weitere Möglichkeit an:

$$b(t) = \frac{1}{\pi} \cdot \frac{\sigma}{\sigma^2 + (t-\theta)^2}, \tag{4.19}$$

wobei σ die Breite und θ den Ort der Lorentzfunktion angibt. Auf diese soll hier aber nicht näher eingegangen werden.

Gaußfunktionen als Basissystem

Zunächst werden die Prototypen als Überlagerungen von gewichteten Gaußfunktionen

$$b_i(t) = \frac{1}{\sqrt{2\pi}\cdot\sigma_i} e^{-\frac{(t-\theta_i)^2}{2\cdot\sigma_i^2}} \tag{4.20}$$

betrachtet, wobei $t \in [1,n]$, $\sigma_i \in \mathbb{R}^+$ die Breite und $\theta_i \in [1,n]$ den Ort der Gaußfunktion angibt. Mit anderen Worten, die Prototypen ergeben sich nun aus einer Linearkombination von $G \in \mathbb{N}$ und $G < n$ gewichteten Gaußfunktionen

$$w(t) = \sum_{i=1}^{G} \beta_i \cdot b_i(t) \tag{4.21}$$

mit $\beta_i \in \mathbb{R}$ die Höhe der i-ten Gaußfunktion. Die Abbildungen 4.1 illustrieren die ange-

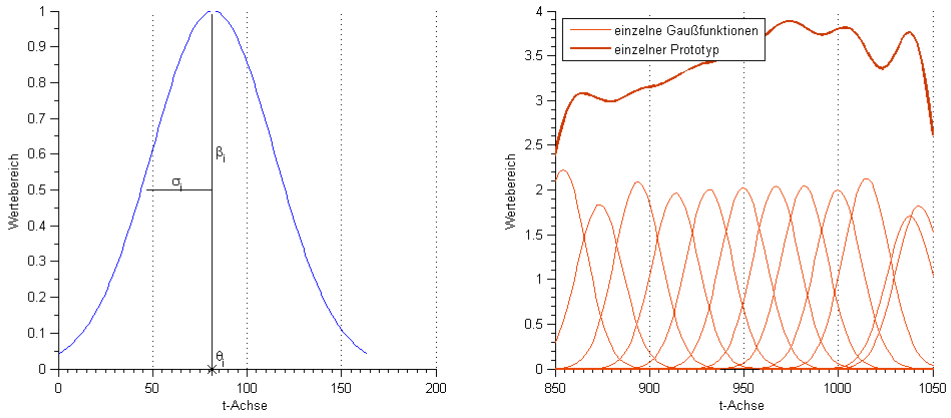

Abbildung 4.1: links: einzelne Gaußfunktion mit Parametern ;rechts: Überlagerung von Gaußfunktionen

gebenen Parameter der Gaußfunktion und deren Überlagerungen zur Darstellung der einzelnen Prototypen. Die dazugehörige partielle Ableitung der Prototypen nach dem Parameter t, welche für die Sobolev-Distanz (4.13) verwendet wird, ist von der Form

$$\dot{w}(t) = -\sum_{i=1}^{G} \beta_i \cdot \frac{t-\theta_i}{\sqrt{2\pi}\cdot\sigma_i^3} e^{-\frac{1}{2}\left(\frac{t-\theta_i}{\sigma_i}\right)^2}. \tag{4.22}$$

Demnach ergibt sich für die Sobolev-Distanz (4.13) zu

$$d_S(\mathbf{x},\mathbf{w},\alpha) = (1-\alpha)\cdot\sum_{t=1}^{n}\left(x(t)-\sum_{i=1}^{G}\beta_i\cdot\frac{1}{\sqrt{2\pi}\cdot\sigma_i}e^{-\frac{1}{2}\left(\frac{t-\theta_i}{\sigma_i}\right)^2}\right)^2 + \alpha\cdot\sum_{t=1}^{n}\left(\dot{x}(t)+\sum_{i=1}^{G}\beta_i\cdot\frac{t-\theta_i}{\sqrt{2\pi}\cdot\sigma_i^3}e^{-\frac{1}{2}\left(\frac{t-\theta_i}{\sigma_i}\right)^2}\right)^2. \tag{4.23}$$

Die Anzahl der verwendeten Gaußfunktionen sollte hinsichtlich der Bemerkung 4.8 so gewählt werden, dass der oben angesprochene Vorteil zur Parameterreduzierung nicht verloren geht.
Für die Adaption der Prototypen werden nicht alle Parameter der Gaußfunktionen gelernt. Bei einer hohen Anzahl von Parametern besteht die Gefahr, dass sich beispielsweise die Höhen β_i und die Breiten σ_i in der Gesamtheit der Linearkombination (4.21) in jedem Lernschritt zu sehr beeinflussen. Das wiederum führt zu einer Instabilität des Verfahrens. Beobachtet wurde dieser Effekt bereit bei RBF-Netzen [18]. Aus diesem Grund werden für die Gaußfunktionen $b_i(t)$ nur die Parameter β_i und θ_i gelernt und die Parameter σ_i für $i = 1,...,G$ geeignet initialisiert und während des Lernprozesses konstant gehalten.

Bemerkung 4.9 Die Dimensionen der Daten werden auf die Abszissenachse übertragen und diese in G äquidistante Intervalle eingeteilt. Die Breiten $\sigma_i,\ i = 1,...,G$ erhalten bei der Initialisierung alle den gleichen konstanten Wert, der zwei Drittel der jeweiligen Intervalllänge beträgt [6].

Anders als bei den Update-Regeln in Abschnitt 4.2, wo die Prototypen in jeder Komponente adaptiert wurden, werden nun die Parameter β_i und θ_i durch einen stochastischen Gradientenabstieg (4.24),(4.25),(4.26) und (4.27) determiniert. Diese wiederum adaptieren die Form der Gaußfunktionen und somit die Lage der Prototypen.
Für die Update-Regeln zur Minimierung der Kostenfunktion (4.2) werden die Höhen und Orte der Gaußfunktionen des Gewinnerprototypen $\mathbf{w}^+$ mit β_i^+ und θ_i^+ bezeichnet. Analog für den Verliererprototypen $\mathbf{w}^-$ mit β_i^- und θ_i^-. Es ergeben sich für den GLVQ nun die folgenden Update-Regeln für die Parameter $\beta_i^\pm$ und $\theta_i^\pm$:

$$\beta_i^+ \leftarrow \beta_i^+ - \varepsilon_\beta\frac{\partial f}{\partial\mu}\cdot\frac{d_S^-(\mathbf{x})}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2}\cdot\frac{\partial d_S^+(\mathbf{x})}{\partial\beta_i^+} \tag{4.24}$$

$$\theta_i^+ \leftarrow \theta_i^+ - \varepsilon_\theta\frac{\partial f}{\partial\mu}\cdot\frac{d_S^-(\mathbf{x})}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2}\cdot\frac{\partial d_S^+(\mathbf{x})}{\partial\theta_i^+} \tag{4.25}$$

$$\beta_i^- \leftarrow \beta_i^- + \varepsilon_\beta\frac{\partial f}{\partial\mu}\cdot\frac{d_S^+(\mathbf{x})}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2}\cdot\frac{\partial d_S^-(\mathbf{x})}{\partial\beta_i^-} \tag{4.26}$$

$$\theta_i^- \leftarrow \theta_i^- + \varepsilon_\theta\frac{\partial f}{\partial\mu}\cdot\frac{d_S^+(\mathbf{x})}{(d_S^+(\mathbf{x})+d_S^-(\mathbf{x}))^2}\cdot\frac{\partial d_S^-(\mathbf{x})}{\partial\theta_i^-} \tag{4.27}$$

Unter der Verwendung der Sobolev-Metrik (4.23) ändern sich die Gradienten $\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \cdot}$ zu:

$$\frac{\partial d_S^{\pm}}{\partial \theta_i^{\pm}} = -\frac{\beta_i^{\pm}}{\sqrt{2\pi}\sigma_i^{\pm 3}} \cdot \left[(1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} \left(\left(x(t) - w^{\pm}(t)\right) \cdot (t-\theta_i^{\pm}) \cdot e^{-\frac{1}{2}\left(\frac{t-\theta_i^{\pm}}{\sigma_i^{\pm}}\right)^2} \right) \right.$$
$$\left. + \alpha \cdot 2 \cdot \sum_{t=1}^{n} \left((\dot{x}(t) - \dot{w}^{\pm}(t)) \cdot \left(\frac{(t-\theta_i^{\pm})^2}{\sigma_i^{\pm}} - 1 \right) \cdot e^{-\frac{1}{2}\left(\frac{t-\theta_i^{\pm}}{\sigma_i^{\pm}}\right)^2} \right) \right]$$

$$\frac{\partial d_S^{\pm}}{\partial \beta_i^{\pm}} = \frac{1}{\sqrt{2\pi} \cdot \sigma_i^{\pm}} \left[(1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} \left((w^{\pm}(t) - x(t)) e^{-\frac{1}{2}\left(\frac{t-\theta_i^{\pm}}{\sigma_i^{\pm}}\right)^2} \right) \right.$$
$$\left. + \frac{2 \cdot \alpha}{\sigma_i^{\pm 2}} \sum_{t=1}^{n} \left((\dot{x}(t) - \dot{w}^{\pm}(t))(t-\theta_i^{\pm}) e^{\frac{1}{2}\left(\frac{t-\theta_i^{\pm}}{\sigma_i^{\pm}}\right)^2} \right) \right]$$

Bemerkung 4.10 Der Gradient der Kostenfunktion bezüglich α wird analog zu Abschnitt 4.2 gebildet.

Die Anwendung von Gaußfunktionen als Basissystem ist vor allem zur Darstellung von Datenvektoren und somit deren Prototypen geeignet, die viele ausgeprägte lokale Minima/Maxima haben. Die wesentlichen Schritte des GLVQ mit Sobolev-Metrik und der Überlagerung von Basisfunktionen sind in dem folgenden Algorithmus aufgezeigt.

Algorithm 3 GLVQ mit Sobolev-Metrik und der Überlagerung von Gaußfunktionen

Eingabe: Anzahl der Gaußfunktionen $G \in \mathbb{N}$, Lernraten ε^{α}, ε^{β} und ε^{θ}, Trainingsdatensatz $\mathscr{T}$, Startwert α, maximale Anzahl der Iterationen k_{max} und Anzahl der Prototypen pro Klasse
Ausgabe: Prototypen
Ermittle geglättete Datenpunkte $\overline{\mathbf{x}}$
Ermittle erste Ableitung der geglätteten Datenpunkte $\dot{\overline{\mathbf{x}}}$
zufällige Initialisierung von G Gaußfunktionen zur Darstellung der Prototypen $\mathbf{w}$
Setze k:=0
while $k < k_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Ermittle Ableitung der Prototypen $\dot{\mathbf{w}}$ nach (4.22)
  Ermittle euklidischen Abstand zwischen allen Prototypen $\mathbf{w}$ und dem Datenpunkt $\mathbf{x}$
  Ermittle euklidischen Abstand zwischen Ableitungen der Prototypen $\dot{\mathbf{w}}$ und dem geglätteten Datenpunkt $\dot{\overline{\mathbf{x}}}$
  Berechne Abstände aller Prototypen zum Datenpunkt $\mathbf{x}$ mit (4.23)
  Ermittle Gewinner- und Verliererprototyp von Datenpunkt $\mathbf{x}$
  Verschiebe Gewinner- und Verliererprototyp durch Adaption der Gaußfunktionen nach (4.24), (4.25) und (4.26), (4.27)
  Adaptiere den Parameter α mit Gleichung (4.17)

$k = k + 1$
end while

Es wird für den obigen Algorithmus nochmals erwähnt, dass die Breiten σ_i der Gaußfunktionen zufällig initialisiert werden und über den Lernprozess konstant bleiben. Im folgenden Verlauf wird ein Ansatz zur Darstellung der Prototypen glatter Datenvektoren aufgezeigt.

Sigmoidfunktionen als Basissystem

Zur Darstellung nicht welliger Prototypen für entsprechende Daten sind Sigmoidfunktionen

$$b_i(t) = \frac{1}{1 + e^{-\frac{(t-\theta_i)}{2\sigma_i^2}}} + \tilde{s}_i \tag{4.28}$$

als Basisfunktionen gut geeignet, wobei der Parameter $\theta_i \in [1, n]$ die Stelle des größten Anstieges angibt. Des Weiteren steht σ_i für die Breite und $\tilde{s}_i$ ist die Verschiebung der Basisfunktion in Ordinatenrichtung. Es ist prinzipiell möglich die Form einer Gaußfunktion durch zwei Sigmoidfunktionen darzustellen (siehe Abbildung 4.2). Aus diesem

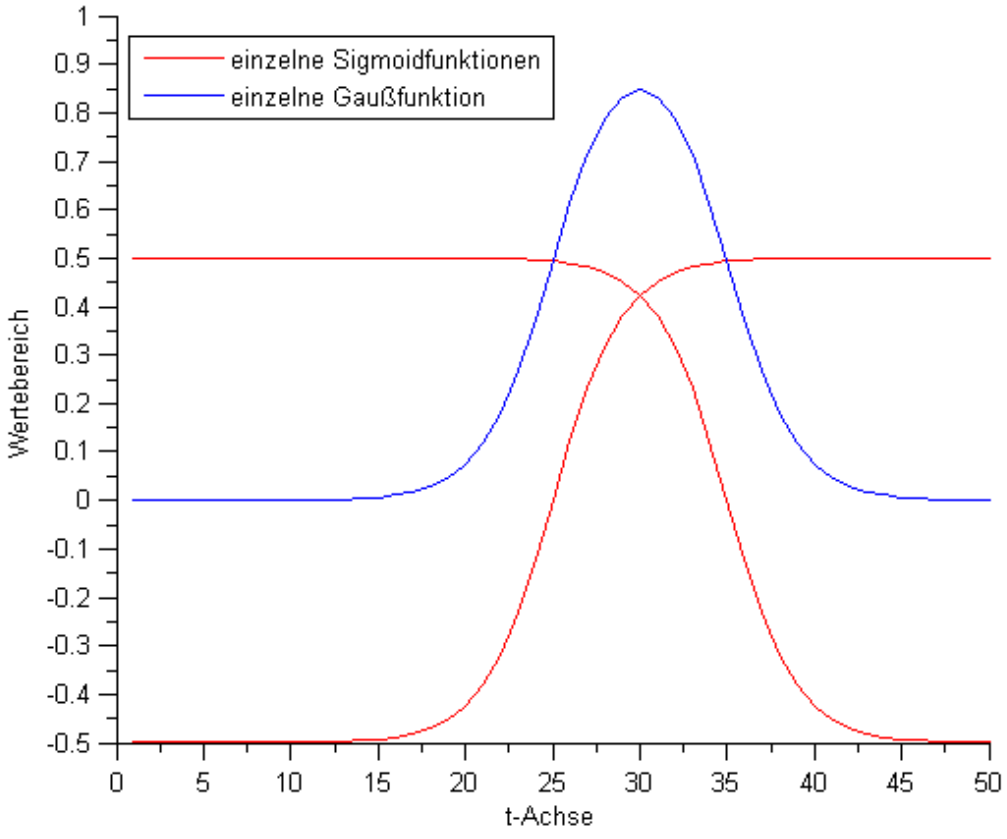

Abbildung 4.2: Gaußfunktion durch Überlagerung zweier Sigmoidfunktionen

Grund können sowohl gewellte als auch nicht gewellte Datenverläufe nachgebildet werden. Man erreicht somit eine größere Flexibilität in der Darstellung der funktionalen Datenvektoren. Damit ergibt sich für die Darstellung der Prototypen durch $G \in \mathbb{N}$ Sigmoidfunktionen der Ausdruck

$$w(t) = \sum_{i=1}^{G} \left[\frac{\beta_i}{1 + e^{-\frac{(t-\theta_i)}{2\sigma_i^2}}} + s_i \right] \tag{4.29}$$

mit der partiellen Ableitung nach t:

$$\dot{w}(t) = \sum_{i=1}^{G} \left[\frac{\beta_i \cdot e^{-\frac{(t-\theta_i)}{2\sigma_i^2}}}{2\sigma_i^2 \left(1 + e^{-\frac{(t-\theta_i)}{2\sigma_i^2}} \right)} \right], \tag{4.30}$$

wobei $\beta_i \in \mathbb{R}$ die Höhe der i-ten Sigmoidfunktion angibt und $s_i = \beta_i \cdot \tilde{s}_i$ ist. In der Abbildung 4.3 wird eine Überlagerung von Sigmoidfunktionen dargestellt und die einzelnen Parameter illustriert. Basierend auf einem stochastischen Gradientenabstieg, werden

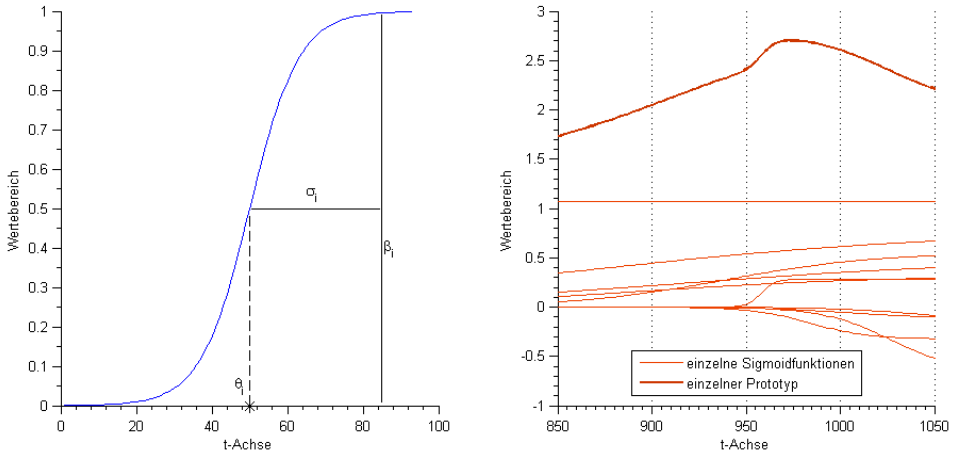

Abbildung 4.3: links: einzelne Sigmoidfunktion mit Parametern; rechts: Überlagerung von Sigmoidfunktionen

die Update-Regeln des GLVQ mit der Sobolev-Metrik und der Überlagerung von Sigmoidfunktionen für die Parameter $\beta_i^{\pm}$,$\sigma_i^{\pm}$,$\theta_i^{\pm}$ und $s_i^{\pm}$ ermittelt. Dabei werden die gleichen Bezeichnungen verwendet, wie in dem vorherigen Abschnitt zu den Gaußfunktio-

nen:

$$\beta_i^{\pm} \leftarrow \beta_i^{\pm} - \varepsilon_r^{\beta} \frac{\partial f}{\partial \mu} \cdot \frac{\partial \mu}{\partial d_S^{\pm}} \cdot \frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \beta_i^{\pm}} \tag{4.31}$$

$$\theta_i^{\pm} \leftarrow \theta_i^{\pm} - \varepsilon_r^{\theta} \frac{\partial f}{\partial \mu} \cdot \frac{\partial \mu}{d_S^{\partial \pm}} \cdot \frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \theta_i^{\pm}} \tag{4.32}$$

$$\sigma_i^{\pm} \leftarrow \sigma_i^{\pm} - \varepsilon_r^{\sigma} \frac{\partial f}{\partial \mu} \cdot \frac{\partial \mu}{\partial d_S^{\pm}} \cdot \frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \sigma_i^{\pm}} \tag{4.33}$$

$$s_i^{\pm} \leftarrow s_i^{\pm} - \varepsilon_r^{s} \frac{\partial f}{\partial \mu} \cdot \frac{\partial \mu}{\partial d_S^{\pm}} \cdot \frac{\partial d_S^{\pm}(\mathbf{x})}{\partial s_i^{\pm}} \tag{4.34}$$

Zur Darstellung der Gradienten der Sobolev-Distanz definiert man zunächst die Funktionen

$$g_i^{\pm}(t) := g(t, \sigma_i^{\pm}, \theta_i^{\pm}) = e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}$$
$$f_i^{\pm}(t) := f(t, \sigma_i^{\pm}, \theta_i^{\pm}) = 1 + g(t, \sigma_i^{\pm}, \theta_i^{\pm}).$$

Somit ergeben sich für die Gradienten der Sobolev-Distanz

$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial s_i^{\pm}} = -(1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} (x(t) - w^{\pm}(t))$$

$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \theta_i^{\pm}} = (1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} \left[(x(t) - w^{\pm}(t)) \cdot \left(\frac{\beta_i^{\pm} \cdot g_i^{\pm}(t)}{2\sigma_i^{\pm 2} \left(f_i^{\pm}(t)\right)^2} \right) \right]$$
$$- \alpha \cdot 2 \cdot \sum_{t=1}^{n} \left[(\dot{x}(t) - \dot{w}^{\pm}(t)) \left(\frac{\beta_i^{\pm} g_i^{\pm}(t) \left(1 - g_i^{\pm}(t)\right)}{4\sigma_i^{\pm 4} \left(f_i^{\pm}(t)\right)^3} \right) \right]$$

$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \beta_i^{\pm}} = -(1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} \left[(x(t) - w^{\pm}(t)) \left(\frac{1}{f_i^{\pm}(t)} \right) \right]$$
$$- \alpha \cdot 2 \cdot \sum_{t=1}^{n} \left[(\dot{x}(t) - \dot{w}^{\pm}(t)) \left(\frac{g_i^{\pm}(t)}{2\sigma_i^{\pm 2} \left(f_i^{\pm}(t)\right)^3} \right) \right]$$

$$\frac{\partial d_S^{\pm}(\mathbf{x})}{\partial \sigma_i^{\pm}} = (1-\alpha) \cdot 2 \cdot \sum_{t=1}^{n} \left[(x(t) - w^{\pm}(t)) \frac{\beta_i (t - \theta_i) g_i^{\pm}(t)}{\sigma_i^3 \left(f_i^{\pm}(t)\right)^2} \right]$$
$$- \alpha \cdot 2 \cdot \sum_{t=1}^{n} \left[(\dot{x}(t) - \dot{w}^{\pm}(t)) \frac{\partial \dot{\mathbf{w}}^{\pm}}{\partial \sigma_i^{\pm}} \right]$$

wobei

$$\frac{\partial \dot{\mathbf{w}}^{\pm}}{\partial \sigma_i^{\pm}} = \frac{\beta_i^{\pm} g_i^{\pm}(t) \left[(t - \theta_i^{\pm}) f_i^{\pm}(t) - 2(\sigma_i^{\pm} f_i^{\pm}(t) + \frac{(t-\theta_i^{\pm})}{\sigma_i^{\pm}} g_i^{\pm}(t)) \right]}{2\sigma_i^{\pm 4} \left[f_i^{\pm}(t) \right]^3}.$$

Es werden die wesentlichen Schritte für den GLVQ mit der Sobolev-Metrik und der Überlagerung von Sigmoidfunktionen in dem folgenden Algorithmus zusammengefasst.

Algorithm 4 GLVQ mit Sobolev-Metrik und der Überlagerung von Sigmoidfunktionen

Eingabe: Anzahl der Sigmoidfunktionen $G \in \mathbb{N}$, Lernraten ε^{α}, ε^{β}, ε^{σ}, ε^{s} und ε^{θ}, Trainingsdatensatz $\mathscr{T}$, Startwert α, maximale Anzahl der Iterationen r_{max} und Anzahl der Prototypen pro Klasse
Ausgabe: Prototypen
Ermittle geglättete Datenpunkte $\overline{\mathbf{x}}$
Ermittle erste Ableitung der geglätteten Datenpunkte $\dot{\overline{\mathbf{x}}}$
zufällige Initialisierung von G Sigmoidfunktionen zur Darstellung der Prototypen $\mathbf{w}$
Setze r:=0
while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Ermittle Ableitung der Prototypen $\dot{\mathbf{w}}$ nach (4.30)
  Ermittle euklidischen Abstand zwischen allen Prototypen $\mathbf{w}$ und dem Datenpunkt $\mathbf{x}$
  Ermittle euklidischen Abstand zwischen Ableitungen der Prototypen $\dot{\mathbf{w}}$ und dem geglätteten Datenpunkt $\dot{\overline{\mathbf{x}}}$
  Berechne Abstände aller Prototypen zum Datenpunkt $\mathbf{x}$
  Ermittle Gewinner- und Verliererprototyp von Datenpunkt $\mathbf{x}$
  Verschiebe Gewinner- und Verliererprototyp durch Adaption der Sigmoidfunktionen nach (4.31), (4.32), (4.33) und (4.34)
  Adaptiere den Parameter α mit Gleichung (4.17)
  $r = r + 1$
end while

4.3 Ergebnisse mit realen Daten

Bisher wurden in diesem Kapitel der GLVQ und verschiedene Modifikationen vorgestellt. Diese Verfahren wurden an den Datensätzen aus Abschnitt 2 getestet und miteinander verglichen. Im Weiteren erhalten die Verfahren folgende Bezeichnungen:

- GLVQ aus Abschnitt 4.1 (Bez.: GLVQ)
- GLVQ mit Sobolev-Metrik aus Abschnitt 4.2 (Bez.: GLVQ+Sobolev)
- GLVQ mit Sobolev-Metrik und Überlagerung von Gaußfunktionen aus Abschnitt 4.2.2 (Bez.: GLVQ+Sobolev+Gauß)
- GLVQ mit Sobolev-Metrik und Überlagerung von Sigmoidfunktionen aus Abschnitt 4.2.2 (Bez.: GLVQ+Sobolev+Sigmoid)

Jedes der vier Verfahren wurde mit 8000 Lernzyklen auf dem Tecator- und Kaffee-Datensatz getestet. Ein Lernzyklus ergibt sich, in dem jeder Datenvektor genau einmal aus einer zufälligen Reihenfolge zur Adaption der Prototypen ausgewählt wurde. Zur

Approximation des relativen Fehlers in der Kostenfunktion (4.2) bzw. (4.14) wurde die Identitätsfunktion verwendet. Man initialisierte pro Klasse einen Prototypen.

Tecator-Datensatz

Die Anzahl der Basisfunktionen sollte hierbei günstig gewählt werden. Bei einer zu geringen Anzahl von Basisfunktionen weisen die Prototypen zu große Unebenheiten in ihren Verläufen auf. Wird die Anzahl zu groß gewählt, kommt es zu der Problematik die bereits in Bemerkung 4.8 genannt wurde. Für die Experimente wurden daher 12 Gaußfunktionen zur Darstellung der Prototypen bei dem Verfahren GLVQ+Sobolev+Gauß und 10 Sigmoidfunktionen bei dem Verfahren GLVQ+Sobolev+Sigmoid verwendet.
Die Startwerte für den Parameter α, der die Relevanz des Ableitungsterms in der Sobolev-Metrik (4.13) angibt, beträgt bei den Tests für alle Verfahren 0.5. Außerdem wird α mit der Lernrate $\varepsilon^{\alpha} = 0.00005$ gelernt. Es ist notwendig, dass die Lernrate für den Parameter α sehr klein gewählt wird (siehe Bemerkung 4.7). Das Lernen des Parameters α darf die Adaption der Prototypen nicht beeinflussen, weil sonst eine Konvergenz des Verfahrens nicht gewährleistet ist. Weitere Eingabeparameter für die Analysen der Verfahren sind in der Tabelle 4.1 angegeben. Hierbei steht ε für die Lernrate der Prototypen

Verfahren	ε	ε^{β}	ε^{σ}	ε^{θ}	ε^{s}
GLVQ	0.01	–	–	–	–
GLVQ+Sobolev	0.01	–	–	–	–
GLVQ+Sobolev+Gauß	–	0.01	–	0.005	–
GLVQ+Sobolev+Sigmoid	–	0.01	0.005	0.005	0.005

Tabelle 4.1: Eingabeparameter für Analysen

und ε^{β}, ε^{σ}, ε^{θ} und ε^{s} für die Lernrate der Höhe, Breite, Ort und der Verschiebung in y-Richtung der Basisfunktionen.
Zunächst werden die Resultate der Prototypen jedes Verfahrens in Abbildung (4.4) betrachtet. Die Prototypen vom GLVQ liegen weiter auseinander als die Prototypen vom GLVQ+Sobolev. Diese Beobachtung lässt sich durch die unterschiedlichen Metriken erklären: Auf Grund der stark überlappenden Klassen des Tecator-Datensatzes gelingt es dem GLVQ mit der euklidischen Metrik nur schwer, die Datenvektoren innerhalb der Überlappung zu klassifizieren. Lediglich die äußeren Datenvektoren kann der GLVQ voneinander trennen. Aus diesem Grund liegen die Prototypen am Rand der Überlappung. Der GLVQ+Sobolev hingegen erkennt auf Grund des Ableitungsterms die Klassenunterschiede innerhalb der Überlappung. Diese Zusatzinformation über die Anstiege der Datenvektoren ermöglicht dem GLVQ+Sobolev die Prototypen enger zusammen zu legen.
Die Form der Prototypen des GLVQ+Sobolev+Gauß weist eine sehr wellige Form auf. Diese Beobachtung liegt nicht an einer zu geringen Anzahl von Gaußfunktionen,[5] son-

[5] Tests mit mehr als 12 Gaußfunktionen zeigten das gleiche Merkmal auf.

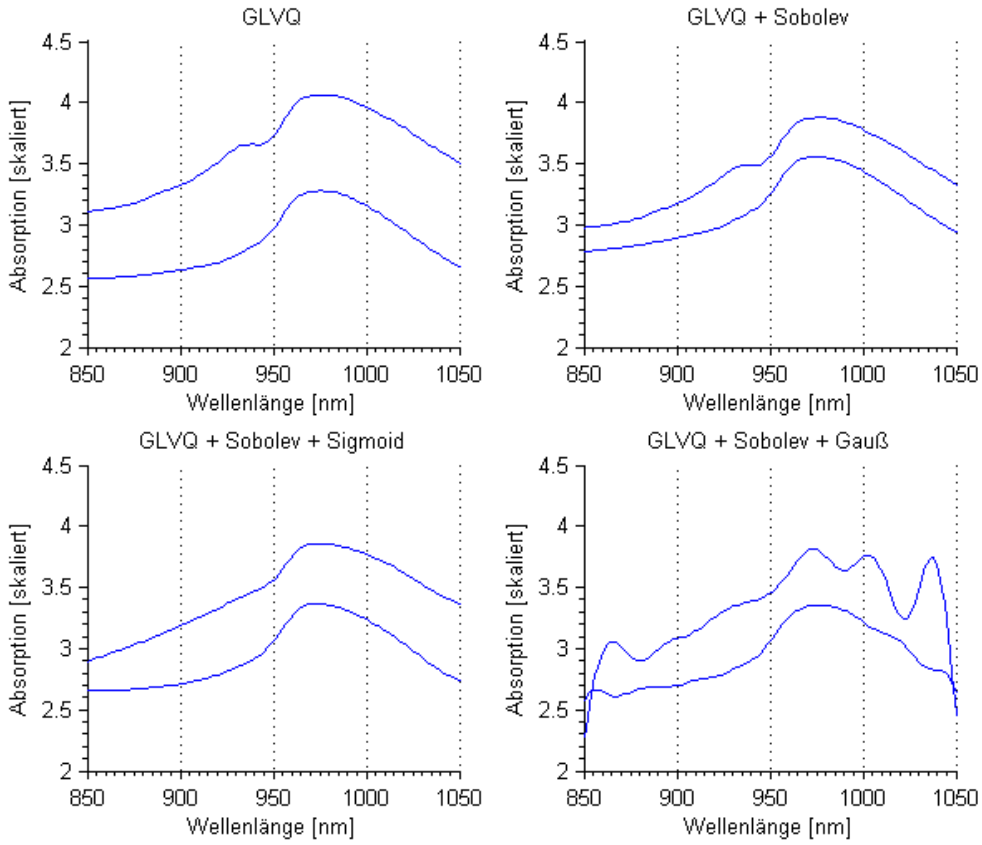

Abbildung 4.4: Prototypen für Tecator-Datensatz

dern diese Art von Basisfunktionen ist zur Darstellung von Prototypen mit einer welligen Form ungeeignet. Vielmehr können Gaußfunktionen für wellige Datenformen verwendet werden.
Die Anwendung von Sigmoidfunktionen zur Darstellung von Prototypen mit glatten Formen erweist sich hingegen für geeignet, wie die Abbildung 4.4 bei dem GLVQ+Sobolev+Sigmoid verdeutlicht. Ein Nachteil ist, dass das lokale Maximum in dem Wellenlängenbereich 920-950nm der Klasse 'Fettreich' nach 8000 Lernzyklen bei diesem Prototyp noch nicht ausgebildet wurde. Bei Tests mit 10000 Lernzyklen hingegen war dieses lokale Maximum deutlicher zu erkennen. Zusätzlich sollte erwähnt werden, dass in den Programmen des GLVQ+Sobolev+Sigmoid und GLVQ+Sobolev+Gauß komplexere Rechnungen durchzuführen sind, was zu einer erhöhten Rechenzeit führte als bei dem GLVQ und GLVQ+Sobolev.
In der Abbildung 4.5 sind nun die Genauigkeitsraten r_a der getesteten Verfahren illustriert. Der GLVQ mit der euklidischen Metrik schafft es 71% der Trainingsdaten und 66% der Testdaten des Tecator-Datensatzes richtig zu klassifizieren. Wendet man den GLVQ mit der Sobolev-Metrik an, ist eine erhebliche Leistungssteigerung zu erkennen. Der GLVQ+Sobolev klassifiziert 95% des Trainingsdatensatzes und 91% des Testdatensatze der Datenvektoren richtig. Dieser Effekt lässt sich auf den Ableitungsterm der Sobolev-Norm zurückführen. Er vergibt die Klassenlabels nicht nur auf Grund des kleinsten eu-

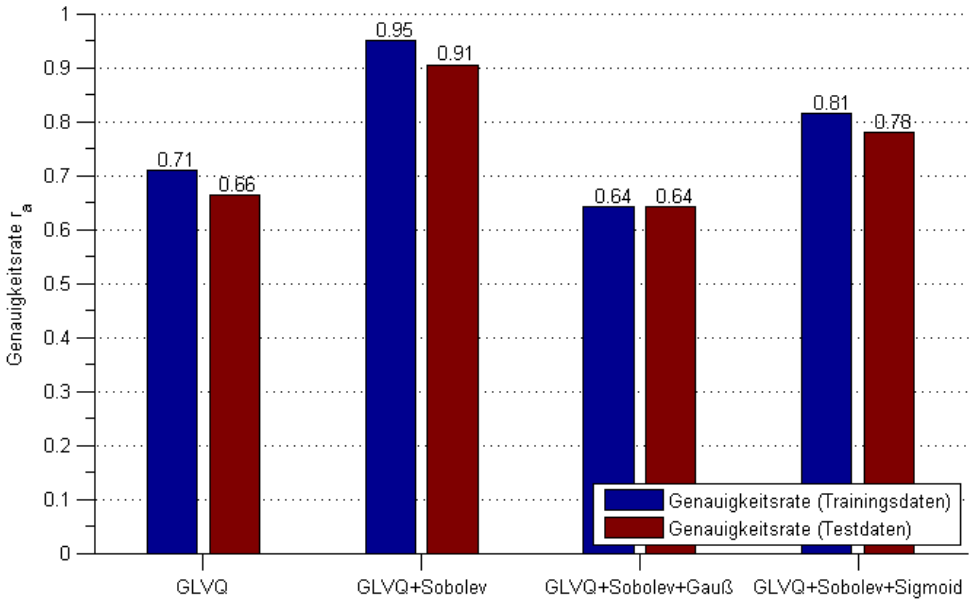

Abbildung 4.5: Genauigkeitsraten

klidischen Abstandes zwischen Prototyp und Datenvektor, sondern der GLVQ+Sobolev bezieht auch Unterschiede in den Ableitungen mit ein. In der Abbildung 4.6 ist der Lernprozess des Parameters α dargestellt mit der dazugehörigen Genauigkeitsrate für den GLVQ+Sobolev.

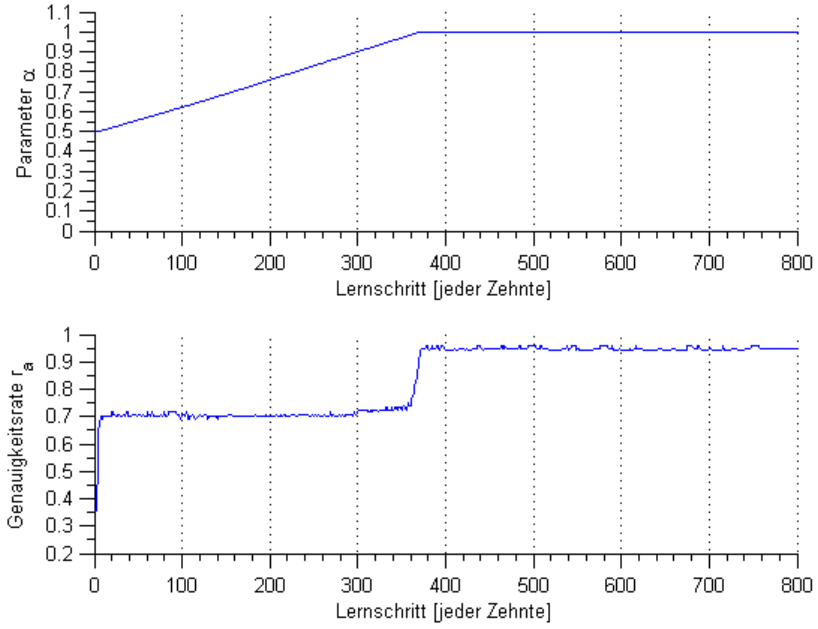

Abbildung 4.6: Einfluss Parameter α auf Genauigkeitsrate (im Verfahren GLVQ+Sobolev)

Der Parameter α wächst während des Lernprozesses an, bis er den Wert ≈ 1 erreicht.

Die Genauigkeitsrate ist am Anfang auf dem gleichen Niveau wie bei dem GLVQ. Auch bei anwachsendem α ist anfangs kleine Leistungssteigerung bzgl. der Genauigkeitsrate r_a zu erkennen. Allerdings springt r_a plötzlich auf den Wert 0.95 wenn $\alpha = 1$ ist. Demnach ist der Ableitungsterm der Sobolev-Metrik von sehr großer Relevanz. Man kann sogar sagen, dass der GLVQ+Sobolev letztendlich nur mit den Ableitungswerten eine bessere Klassenaufteilung erreicht.
Die Verfahren GLVQ+Sobolev+Gauß/Sigmoid weisen das gleiche Merkmal auf. Beide Verfahren setzen im Verlauf des Lernprozesses $\alpha = 1$. Die Genauigkeitsrate des GLVQ+Sobolev+Gauß wird angesichts dieser Tatsache nicht verbessert. Dafür weisen die Prototypen zu starke Abweichungen von den Datenvektoren des Tecator-Datensatze auf. Für den GLVQ+Sobolev+Sigmoid hingegen ist eine Leistungssteigerung durch α zu erkennen. Die Genauigkeitsrate springt jedoch nicht plötzlich auf einen Wert, wie in Abbildung 4.5, sondern steigt langsam an.
Zusammenfassend zeigten die Ergebnisse eine erhebliche Leistungssteigerung des GLVQ auf dem Tecator-Datensatz durch die Anwendung der Sobolev-Metrik. Des Weiteren wurde nachgewiesen, dass eine Darstellung der Prototypen durch geeignete Basisfunktionen prinzipiell möglich ist und ebenfalls zu guten Ergebnissen führt, wenn die Basisfunktionen geeignet gewählt sind. Dabei muss jedoch ein erhöhter Rechenaufwand in Kauf genommen werden.

Kaffee-Datensatz

Der Kaffee-Datensatz II aus Kapitel 2 weist sehr wellige Verläufe der Datenvektoren in den einzelnen Klassen auf, welche im Wesentlichen die gleichen Anstiege und Trends haben. Außerdem sind die Klassen nicht stark überlagert. Für die Tests des Verfahrens GLVQ+Sobolev+Gauß wurden 20 Gaußfunktionen und für das Verfahren GLVQ+Sobolev+Sigmoid 32 Sigmoidfunktionen verwendet. Die Anzahl der Basisfunktionen soll dabei nicht zu klein und nicht zu groß gewählt werden, wie schon bei dem Tecator-Datensatz erwähnt. Die Startwerte des Parameters α wurden stets mit 0.3, 0.5 und 0.7 gewählt, wobei im Folgenden zu erkennen sein wird, dass diese Änderungen nicht ausschlaggebend für die Testergebnisse sind. Der Relevanz-Faktor α für den Ableitungsterm in der Sobolev-Metrik wurde mit der Lernrate $\varepsilon^{\alpha} = 0.00005$ gelernt. Außerdem wurden die weiteren Parameter der Basisfunktionen mit den Lernraten aus Tabelle 4.1 gelernt. Betrachtet man als erstes die Verläufe der Prototypen in Abbildung 4.7, ist zwischen dem Verfahren GLVQ und GLVQ+Sobolev kaum ein wesentlicher Unterschied im Verlauf der Prototypen zu erkennen. Beide Verfahren bilden sowohl Minima- als auch Maximaverläufe detailliert aus. Das spricht für die Qualität der beiden Verfahren. Auch die Prototypen der Verfahren GLVQ+Sobolev+Gauß und GLVQ+Sobolev+Sigmoid ähneln sich in ihrer Darstellung. Im Vergleich zum GLVQ und GLVQ+Sobolev sind die Prototypen nicht so detailliert ausgeprägt. Vielmehr weisen diese eine glatte Form auf.

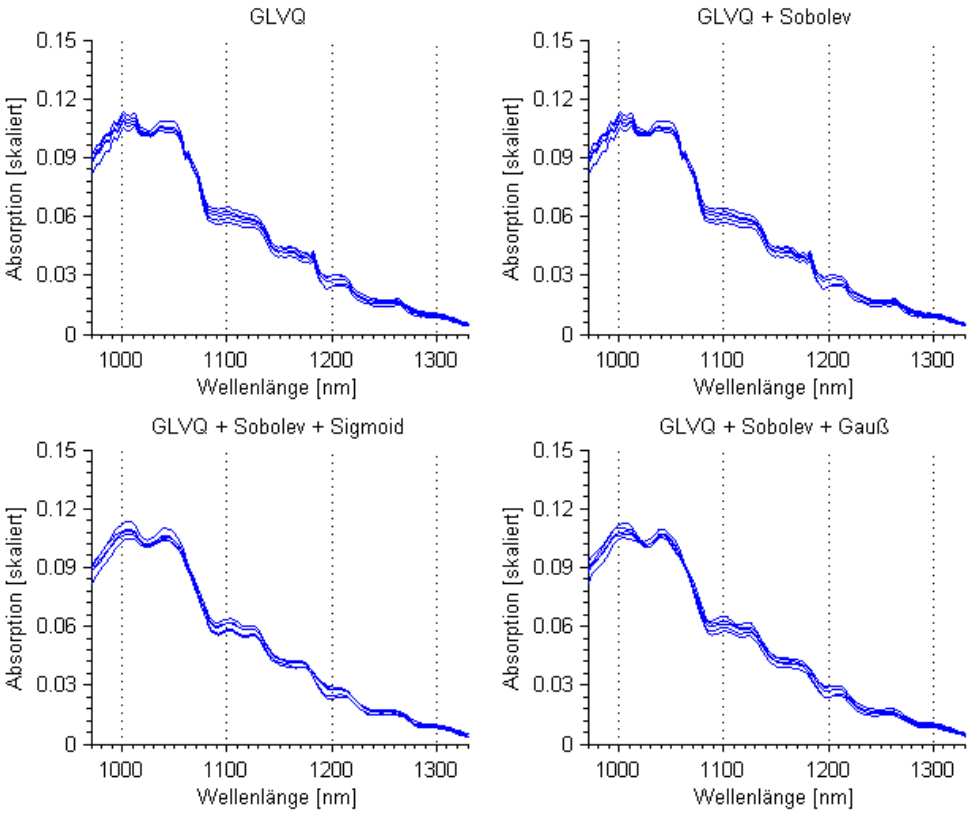

Abbildung 4.7: Prototypen für den Kaffee-Datensatz

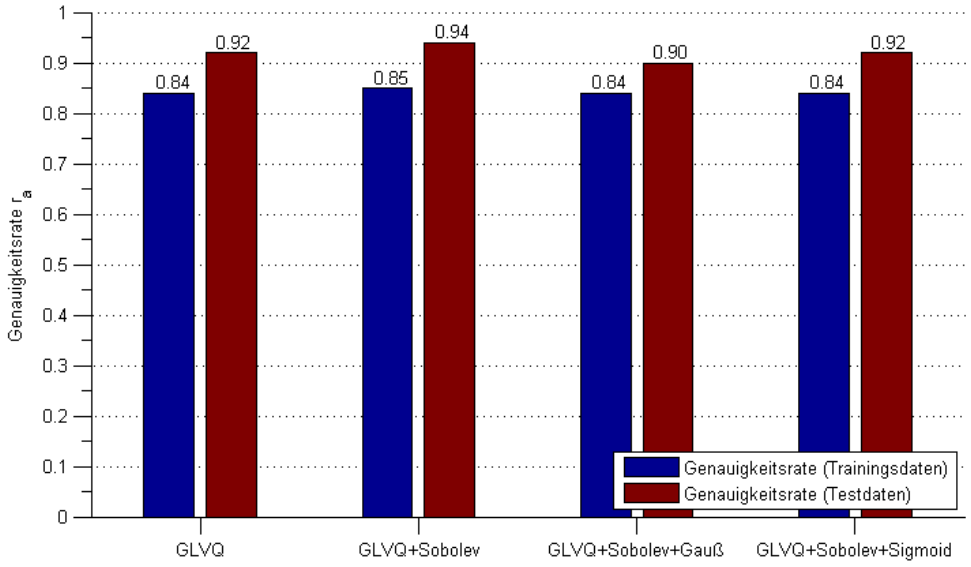

Abbildung 4.8: Genauigkeitsrate der einzelnen Verfahren für den Kaffee-Datensatz

Hinsichtlich der Genauigkeitsraten r_a der einzelnen Verfahren in Abbildung 4.8 bringt diese glatte Darstellung der Prototypen aber keinen Nachteil. Alle Verfahren klassifizie-

ren im Wesentlichen den gleichen Anteil an Test- und Trainingsdaten richtig. Dabei ist eine detaillierte Darstellung der Prototypen nicht notwendig, wie beispielsweise in den Ergebnissen des GLVQ+Sobolev und GLVQ+Sobolev+Gauß zu erkennen ist. Hinsichtlich dieser Tatsache lässt sich für den Kaffee-Datensatz folgende Aussage machen: Hinter dem zackenförmigen Verlauf der einzelnen Datenvektoren befinden sich keine relevanten Informationen die zur Trennung der Klassen verwendet werden können. Vielmehr handelt es sich hierbei um ein unrelevantes Rauschen. Es erreichen daher glatte Prototypen das gleich Resultat wie detaillierte Prototypen des GLVQ+Sobolev.
In der folgenden Abbildung 4.9 sind nun die Entwicklungen des Parameters α für die einzelnen Verfahren illustriert.

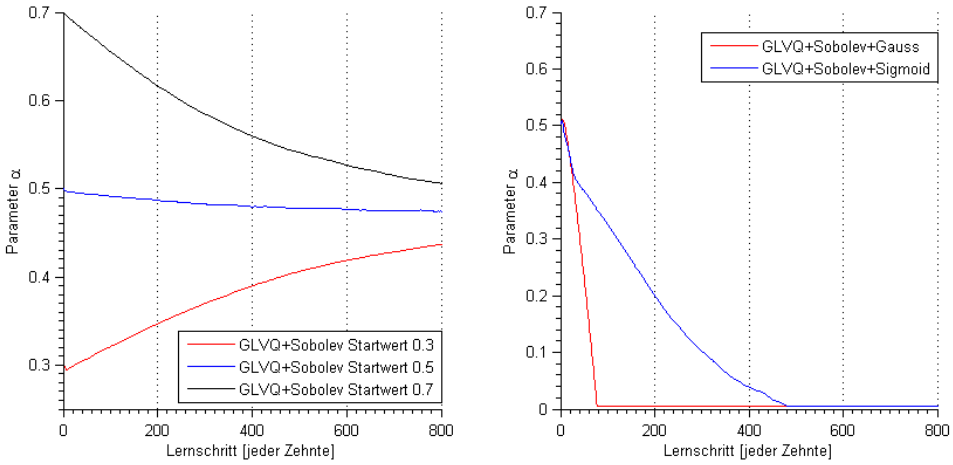

Abbildung 4.9: Entwicklung des Parameters α

In der linken Abbildung (4.9) ist zu erkennen, dass der Parameter α bei dem Verfahren GLVQ+Sobolev gegen den Wert 0.4676 konvergiert, was sich jedoch nicht positiv auf die Genauigkeitsrate auswirkt. In der rechten Abbildung allerdings geht der Parameter α mit dem Startwert 0.5 für die Verfahren GLVQ+Sobolev+Gauß und GLVQ+Sobolev+Sigmoid gegen Null. Diese Beobachtung könnte wie folgt begründet werden: Die detaillierte Ausprägung der Prototypen von dem GLVQ+Sobolev berücksichtigt bei der Entwicklung des Parameters α das oben angesprochene Rauschen der Daten. Der Algorithmus versucht somit Informationen aus diesem Rauschen herauszufiltern und lässt den Parameter α konvergieren. Dagegen die Verfahren GLVQ+Sobolev+Sigmoid/Gauß berücksichtigen das Rauschen nicht, durch die glatte Darstellung der Prototypen. Außerdem sind die groben Anstiege der Datenvektoren bzgl. der einzelnen Klassen im Wesentlichen gleich. Folglich wird α im Lernprozess Null gesetzt, da sowohl kleine als auch große Anstiege keine relevanten Faktoren für die Trennung der einzelnen Klassen darstellen. Jedoch ergaben weitere Tests mit einem geglätteten Kaffee-Datensatz, die gleichen Ergebnisse für GLVQ, GLVQ+Sobolev und GLVQ+Sobolev+Basisfunktionen. Lediglich der Parameter α konvergiert bei dem GLVQ+Sobolev gegen den Wert 0.6314. Für die anderen beiden Verfahren mit Überlagerungen von Basisfunktionen jedoch geht der Parameter α gegen Null. Wären diese kleineren Anstiege

(Rauschen) tatsächlich irrelevant für die Klassifizierung, wäre der Parameter α bei dem GLVQ+Sobolev bei dem geglätteten Datensatz gegen Null gegangen.
Vielmehr liegt die Erklärung in der Form der Kostenfunktionen. Alle Verfahren erreichen auf dem geglätteten und ungeglätteten Kaffee-Datensatz in etwa die gleichen Resultate. Demnach besitzt die zu minimierende Kostenfunktion mehrere lokale Minima mit ungefähr den gleichen Funktionswerten. Je nach Verfahren und der jeweiligen Belegung von α wird das dementsprechende lokale Minimum erreicht. Die folgende Abbildung illustriert den Sachverhalt. Zusammenfassend lässt sich sagen, dass die Sobolev-Metrik

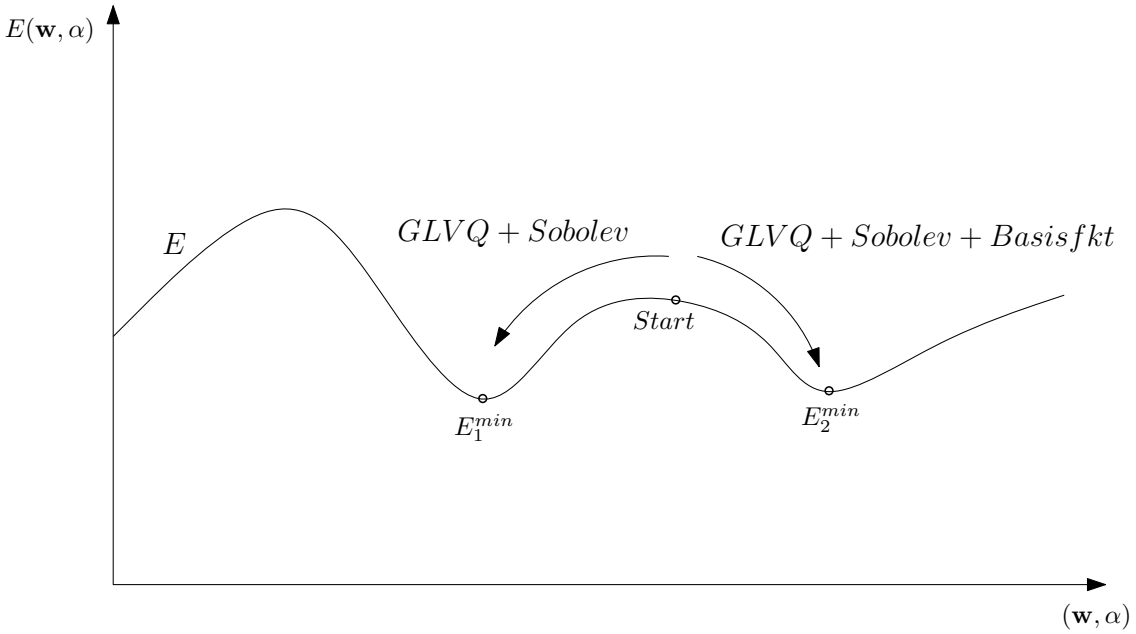

Abbildung 4.10: Schematische Darstellung der Kostenfunktion für den Kaffee-Datensatz (nicht der Originalverlauf der Kostenfunktion)

für den Kaffee-Datensatz keine Verbesserung hinsichtlich der Genauigkeitsrate brachte. Hierfür waren zu geringe Unterschiede in den Verläufen der Datenvektoren. Dennoch wurde gezeigt, dass der GLVQ+Sobolev in der Lage ist den Parameter α konvergieren zu lassen. Durch die Verwendung von Basisfunktionen zur Darstellung der Prototypen erlangte man die Erkenntnis, dass die Kostenfunktion bzgl. das Kaffee-Datensatzes mehrere lokale Minima besitzt, die ähnliche Funktionswerte haben. Aus diesem Grund erreichen die Verfahren GLVQ+Sobolev+Sigmoid/Gauß das gleiche Resultat, wie GLVQ und GLVQ+Sobolev.

5 Generalized Relevance Learning Vectorquantization

5.1 Theorie Generalized Relevance Learning Vector Quantization

Für diesen Abschnitt wurden im Wesentlichen die Quellen [3] und [12] zur Hilfe genommen. 'Generalized Relevance Learning Vector Quantization' (GRLVQ) wurde das erste Mal 2002 von B. Hammer und T. Villmann in einer gleichnamigen Veröffentlichung [3] publiziert. Die Grundidee ist hierbei dieselbe, wie bei dem GLVQ aus Abschnitt 4.1. Man verwendet eine Kostenfunktion, welche die Anzahl der Fehlklassifikationen approximiert. Diese wird durch einen stochastischen Gradientenabstieg minimiert. Zusätzlich sollen nun Informationen über die einzelnen Dimensionen der Datenvektoren gelernt werden, die für die Trennung der Klassen relevant sind. Dieser Informationsgewinn soll bei der Adaption der Prototypen und Berechnung derer Abstände zu den Datenvektor verwendet werden.
Sei nun $\mathscr{T} = \{(\mathbf{x},k) \mid \mathbf{x} \in \mathbb{R}^n,\ k \in \mathscr{K}\}$ die Trainingsdatenmenge mit n-dimensionalen Datenvektoren. Es wird nun für die Berechnung der Abstände zwischen den Prototypen $\mathbf{w}$ und Datenvektoren $\mathbf{x}$ die gewichtete euklidische Metrik

$$d_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w}) = \sum_{t=1}^{n} \left[\lambda(t) \cdot (x(t) - w(t))^2\right] \tag{5.1}$$

vorgestellt. Für die Gewichte gilt $\sum_{t=1}^{n} \lambda(t) = 1$ und $\lambda(t) \geq 0,\ \forall t$. Des Weiteren wird darauf hingewiesen, dass (5.1) eine Quasimetrik ist. Der Gewichtsvektor $\boldsymbol{\lambda}$ beinhaltet in der Komponente $\lambda(t)$ den Wert, der die Relevanz der Dimension t angibt. Durch die zusätzliche Adaption des Relevanzvektors $\boldsymbol{\lambda}$ werden die Abstände relevanter Dimensionen in der euklidischen Metrik stärker gewichtet. Demnach betrachtet man spezifische Merkmale in den Datenvektoren zur Klassentrennung genauer.
In jedem Lernschritt wird nun ein Datenvektor $\mathbf{x}$ zufällig ausgewählt. Für diesen Datenvektor müssen nun der Gewinner- und Verliererprototyp $\mathbf{w}^+$ und $\mathbf{w}^-$ mit den dazugehörigen Abständen $d^+_{\boldsymbol{\lambda}}(\mathbf{x})$ und $d^-_{\boldsymbol{\lambda}}(\mathbf{x})$ ermittelt werden. Mit dieser neuen Metrik (5.1) ändert sich die Funktion (4.1) der relativen Fehlerdistanz zu

$$\mu_{\boldsymbol{\lambda}}(\mathbf{x}) = \frac{d^+_{\boldsymbol{\lambda}}(\mathbf{x}) - d^-_{\boldsymbol{\lambda}}(\mathbf{x})}{d^+_{\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{\boldsymbol{\lambda}}(\mathbf{x})}.$$

Somit wird die Anzahl der Fehlklassifikationen bei dem GRLVQ durch die Kostenfunktion

$$E^{GRLVQ} = \sum_{\mathbf{x}\in\mathcal{T}} f(\mu_{\boldsymbol{\lambda}}(\mathbf{x})) \tag{5.2}$$

approximiert, wobei f eine monoton wachsende Funktion ist. Die Kostenfunktion E^{GRLVQ} wird nun durch einen stochastischen Gradientenabstieg minimiert, indem die Prototypen sukzessive verschoben werden. Zusätzlich soll die Kostenfunktion (5.2) bezüglich $\boldsymbol{\lambda}$ durch einen stochastischen Gradientenabstieg minimiert werden, d.h.:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \varepsilon_r^{\boldsymbol{\lambda}} \frac{\partial E^{GRLVQ}}{\partial \boldsymbol{\lambda}}, \tag{5.3}$$

wobei $\varepsilon_r^{\boldsymbol{\lambda}}$ die Lernrate für den Gewichtsvektor $\boldsymbol{\lambda}$ im r-ten Iterationsschritt angibt. Hierfür muss gelten

$$\varepsilon_r^{\boldsymbol{\lambda}} << \varepsilon_r. \tag{5.4}$$

Der Grund wurde bereits in Bemerkung 4.7 ausführlich angegeben. Dabei steht ε_r für die Lernrate der Prototypen im r-ten Iterationsschritt. Für die Update-Regeln der Prototypen $\mathbf{w}^{\pm}$ und des Relevanzvektors $\boldsymbol{\lambda}$ ergeben sich somit die folgenden Formeln:

$$\mathbf{w}^{+} \leftarrow \mathbf{w}^{+} + \varepsilon_r \frac{\partial f}{\partial \mu_{\boldsymbol{\lambda}}} \cdot \frac{d_{\boldsymbol{\lambda}}^{-}(\mathbf{x}) \cdot \boldsymbol{\lambda} \cdot (\mathbf{x} - \mathbf{w}^{+})}{(d_{\boldsymbol{\lambda}}^{+}(\mathbf{x}) + d_{\boldsymbol{\lambda}}^{-}(\mathbf{x}))^2} \tag{5.5}$$

$$\mathbf{w}^{-} \leftarrow \mathbf{w}^{-} - \varepsilon_r \frac{\partial f}{\partial \mu_{\boldsymbol{\lambda}}} \cdot \frac{d_{\boldsymbol{\lambda}}^{+}(\mathbf{x}) \cdot \lambda \cdot (\mathbf{x} - \mathbf{w}^{-})}{(d_{\boldsymbol{\lambda}}^{+}(\mathbf{x}) + d_{\boldsymbol{\lambda}}^{-}(\mathbf{x}))^2} \tag{5.6}$$

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \varepsilon_r^{\boldsymbol{\lambda}} \frac{\partial f}{\partial \mu_{\boldsymbol{\lambda}}} \left(\frac{d_{\boldsymbol{\lambda}}^{-}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{w}^{+})^2 - d_{\boldsymbol{\lambda}}^{+}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{w}^{-})^2}{(d_{\boldsymbol{\lambda}}^{+}(\mathbf{x}) + d_{\boldsymbol{\lambda}}^{-}(\mathbf{x}))^2} \right) \tag{5.7}$$

Es handelt sich bei dem GRLVQ und GLVQ um das gleiche Grundprinzip. Demnach gelten für den GRLVQ die gleichen Konvergenzkriterien von Kushner und Clark in [7], wie schon bei dem GLVQ. Hierfür wird auf die Bedingungen (4.7), (4.8) und (4.9) verwiesen.

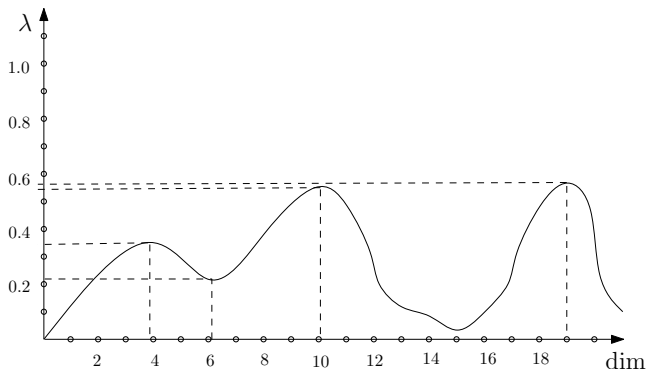

Abbildung 5.1: Beispiel für ein Relevanzprofil

Wie bereits angesprochen, gibt das gelernte Relevanzprofil Aufschluss über die Relevanz einzelnen Dimensionen, die für die Klassentrennung wichtig ist. Die folgende Abbildung (5.1) illustriert diesen Sachverhalt. An dem Relevanzprofils aus Abbildung (5.1) ist zu erkennen, dass beispielsweise die Dimension 4 mit 0.35 erheblich relevanter ist als die Dimensionen 6 mit 0.21 oder 15 mit 0.05. Anhand dieser Erkenntnisse werden die Distanzen von relevanteren Dimensionen in der Metrik automatisch stärker gewichtet. Um die Relevanzen besser interpretieren zu können, sollten diese normiert werden mit

$$\lambda(t) = \frac{\lambda(t)}{\sum_{t=1}^{n} \lambda(t)}, \ \forall t.$$

Die wichtigsten Schritte sind in dem folgenden Algorithmus zusammengefasst.

Algorithm 5 GRLVQ

Eingabe: Trainingsdatensatz $\mathscr{T}$, maximale Anzahl der Iterationen r_{max}, Anzahl der Prototypen pro Klasse und Lernraten ε und $\varepsilon^{\boldsymbol{\lambda}}$
Ausgabe: Prototypen
zufällige Initialisierung der Prototypen $\mathbf{w}$ für jede Klasse
Setze r:=0
while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Berechne alle Abstände der Prototypen $\mathbf{w}$ zu dem Datenpunkt $\mathbf{x}$
  Ermittle den zu $\mathbf{x} \in \mathscr{T}$ gehörigen Gewinner- $\mathbf{w}^+$ und Verliererprototyp $\mathbf{w}^-$
  Adaptiere Gewinner- und Verliererprototyp nach Gleichungen (5.5) und (5.6)
  Adaptiere den Relevanzvektor $\boldsymbol{\lambda}$ mit (5.7)
  $r = r + 1$
end while

5.2 Modifikation des GRLVQ

In diesem Abschnitt wird nun eine Erweiterung des GRLVQ aus Abschnitt 5.1 vorgestellt. Bisher arbeitete der GRLVQ mit der gewichteten euklidischen Metrik (5.1). Dabei stellt $\boldsymbol{\lambda}$ einen Relevanzvektor dar, dessen Komponente $\lambda(t)$ die Relevanz der Datendimension t bei der Klassentrennung repräsentiert.
Seien nun n-dimensionelle Datenvektoren $\mathbf{x} = (x(1), \ldots, x(n))^T$ in dem Datenraum $\mathscr{X}$ und somit auch in der Trainingsdatenmenge $\mathscr{T}$ mit einem funktionalen Zusammenhang gegeben. Das bedeutet, die Komponenten eines Datenvektors werden als Folge von Funktionswerten aufgefasst, die in Abhängigkeit der Zeit $t \in [1,n]$ sind. Der dabei entstehende Verlauf der Datenvektoren weist eine gewisse Glätte und Stetigkeit auf. Aus diesem Grund wird angenommen, dass $\mathscr{T} \subseteq \mathscr{X} \subseteq C^1[1,n]$ ist. Anhand dieses Sachverhaltes, wird nun die Sobolev-Metrik aus Abschnitt 4.2 verwendet. Es wurde bereits gezeigt, dass diese funktionale Metrik für manche Datensätze eine erhebliche Verbes-

serung des GLVQ aufweist. Durch die zusätzliche Betrachtung der Distanzen der ersten Ableitungen erreichte der GLVQ eine bessere Flexibilität bei der Klassentrennung. Im Folgenden wird der GRLVQ dahingehend abgewandelt, dass die Sobolev-Metrik angewendet wird, d.h. es soll nun zusätzlich mit der Sobolev-Metrik ein Relevanzprofil für die einzelnen Dimensionen gelernt werden. Dieses Relevanzprofil bezieht sich nicht nur auf die euklidischen Distanzen zwischen die Datenvektoren und Prototypen, sondern auch auf deren Ableitungen. Hierfür setzt man für die euklidische Metrik $\|\cdot\|_E$ in der Sobolev-Metrik die gewichtete euklidische Metrik (5.1) ein, so dass sich die folgende Quasi-Metrik ergibt:

$$\begin{aligned} d_{S,\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w},\alpha) &= (1-\alpha)\cdot d^2_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w}) + \alpha \cdot d^2_{\boldsymbol{\lambda}}(\dot{\mathbf{x}},\dot{\mathbf{w}}) \\ &= (1-\alpha)\cdot \sum_{t=1}^{n} \left[\lambda(t)\cdot(x(t)-w(t))^2\right] \\ &+ \alpha \sum_{t=1}^{n} \left[\lambda(t)\cdot(\dot{x}(t)-\dot{w}(t))^2\right] \end{aligned} \tag{5.8}$$

Weiter wird $d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) := d_{S,\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w}^+,\alpha)$ als der Gewinnerabstand zwischen Datenvektor $\mathbf{x}$ und Gewinnerprototyp $\mathbf{w}^+$ definiert, wobei der Gewinnerprototyp der am nächstgelegene Prototyp ist, der von der gleichen Klasse ist wie der Datenvektor $\mathbf{x}$. Der Verliererabstand $d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}) := d_{S,\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w}^-,\alpha)$ ist analog definiert, wobei der Verliererprototyp der am nächstgelegene Prototyp ist, der von einer anderen Klasse ist wie der Datenvektor $\mathbf{x}$. Unter der Verwendung der Sobolev-Metrik kann die Anzahl der Fehlklassifikationen des GRLVQ approximiert werden mit der Kostenfunktion

$$E_S^{GRLVQ} = \sum_{\mathbf{x}\in\mathcal{T}} f(\mu_{S,\boldsymbol{\lambda}}(\mathbf{x})), \tag{5.9}$$

wobei

$$\mu_{S,\boldsymbol{\lambda}}(\mathbf{x}) = \frac{d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) - d^-_{S,\boldsymbol{\lambda}}(\mathbf{x})}{d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{S,\boldsymbol{\lambda}}(\mathbf{x})}.$$

Die Kostenfunktion (5.9) soll nun wieder durch einen stochastischen Gradientenabstieg minimiert werden. Hierfür wird in jedem Lernschritt ein Datenvektor $\mathbf{x}$ ausgewählt, an dem dessen Gewinner- und Verliererprototyp $\mathbf{w}^+$ und $\mathbf{w}^-$ adaptiert werden sollen. Für diese Adaption der Prototypen ergeben sich die folgenden Update-Regeln:

$$\mathbf{w}^{\pm} \leftarrow \mathbf{w}^{\pm} - \varepsilon_r \frac{\partial E_S^{GRLVQ}}{\partial \mathbf{w}^{\pm}} \tag{5.10}$$

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \varepsilon_r^{\boldsymbol{\lambda}} \frac{\partial E_S^{GRLVQ}}{\partial \boldsymbol{\lambda}} \tag{5.11}$$

$$\alpha \leftarrow \alpha - \varepsilon_r^{\alpha} \frac{\partial E_S^{GRLVQ}}{\partial \alpha} \tag{5.12}$$

Für die Lernraten im Lernschritt r sollte gelten:

$$\varepsilon_r^{\boldsymbol{\lambda}} << \varepsilon_r^{\alpha} << \varepsilon_r, \ \forall r.$$

Hierbei ist die zweite Ungleichung unmittelbar klar. Da die Metrik stets langsamer gelernt werden muss als die Prototypen (Bemerkung 4.7). Jedoch die erste Ungleichung bedarf einer näheren Erläuterung. Die Summe der Änderungen des Vektors $\boldsymbol{\lambda}$ sind vor allem bei einer großen Anzahl von Datendimension in sehr groß. Wogegen die Änderungen von α bei den meisten Tests kleiner ausfallen. Dieser große Unterschied zwischen den Änderungen bewirkt eine Instabilität bei der Adaption von α. Aus diesem Grund hat es sich als günstig herausgestellt $\boldsymbol{\lambda}$ vor allem bei hohen Datendimensionen viel langsamer gelernt werden als α, um ein möglichst stabiles Lernverhalten zu ermöglichen.
Die Gradienten der Kostenfunktion (5.2) ergeben sich mit der Sobolev-Metrik (5.8) zu

$$\begin{aligned}
\frac{\partial E_S^{GRLVQ}}{\partial \mathbf{w}^+} &= -\frac{\partial f}{\partial \mu_{S,\boldsymbol{\lambda}}} \cdot \frac{d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \boldsymbol{\lambda} \cdot (1-\alpha) \cdot (\mathbf{x}-\mathbf{w}^+)}{(d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}))^2} \\
\frac{\partial E_S^{GRLVQ}}{\partial \mathbf{w}^-} &= \frac{\partial f}{\partial \mu_{S,\boldsymbol{\lambda}}} \cdot \frac{d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \boldsymbol{\lambda} \cdot (1-\alpha) \cdot (\mathbf{x}-\mathbf{w}^-)}{(d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}))^2} \\
\frac{\partial E_S^{GRLVQ}}{\partial \boldsymbol{\lambda}} &= \frac{\partial f}{\partial \mu_{S,\boldsymbol{\lambda}}} \left(\frac{d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \frac{\partial d^+_{S,\boldsymbol{\lambda}}(\mathbf{x})}{\partial \boldsymbol{\lambda}} - d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \frac{\partial d^-_{S,\boldsymbol{\lambda}}(\mathbf{x})}{\partial \boldsymbol{\lambda}}}{(d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}))^2} \right) \\
\frac{\partial E_S^{GRLVQ}}{\partial \alpha} &= \frac{\partial f}{\partial \mu_{S,\boldsymbol{\lambda}}} \left(\frac{d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \frac{\partial d^+_{S,\boldsymbol{\lambda}}(\mathbf{x})}{\partial \alpha} - d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) \cdot \frac{\partial d^-_{S,\boldsymbol{\lambda}}(\mathbf{x})}{\partial \alpha}}{(d^+_{S,\boldsymbol{\lambda}}(\mathbf{x}) + d^-_{S,\boldsymbol{\lambda}}(\mathbf{x}))^2} \right)
\end{aligned}$$

Es wird nun der Algorithmus des GRLVQ unter der Verwendung der Sobolev-Metrik angegeben.

Algorithm 6 GRLVQ mit Sobolev-Norm

Eingabe: Lernraten ε, $\varepsilon^{\boldsymbol{\lambda}}$ und ε^{α}, Trainingsdatensatz $\mathscr{T}$, Startwert α, maximale Anzahl der Iterationen r_{max} und Anzahl der Prototypen pro Klasse
Ausgabe: Prototypen
Ermittle geglättete Datenpunkte $\overline{\mathbf{x}}$
Ermittle erste Ableitung der geglätteten Datenpunkte $\dot{\overline{\mathbf{x}}}$
zufällige Initialisierung der Prototypen $\mathbf{w}$
Setze r:=0
while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Ermittle geglättete Prototypen $\overline{\mathbf{w}}$ und deren Ableitung $\dot{\overline{\mathbf{w}}}$
  Ermittle gewichteten euklidischen Abstand zwischen allen Prototypen $\mathbf{w}$ und dem Datenpunkt $\mathbf{x}$ mit (5.1)

Ermittle gewichteten euklidischen Abstand zwischen allen geglätteten Prototypen $\dot{\tilde{\mathbf{w}}}$ und dem geglätteten Datenpunkt $\dot{\tilde{\mathbf{x}}}$ mit (5.1)
Berechne Abstände aller Prototypen zum Datenpunkt $\mathbf{x}$ mit (5.8)
Ermittle Gewinner- und Verliererprototyp von Datenpunkt $\mathbf{x}$
Adaptiere Gewinner- und Verliererprototyp nach (5.10)
Adaptiere den Parameter α mit (5.12)
Adaptiere den Relevanzvektor $\boldsymbol{\lambda}$ mit (5.11)
$r = r + 1$
end while

In diesem Verfahren kann nun zusätzlich auch der Ansatz eingebaut werden, dass man die Prototypen durch Überlagerungen von Basisfunktionen darstellt. Damit erreicht man auch für den GRLVQ eine Reduktion der zu lernenden Parameter. Außerdem können analytische Ableitungen verwendet werden. In diesem Kapitel wird auf diese Ansatz allerdings nicht eingegangen, weil die Herangehensweise die Gleiche wie in Kapitel 4 ist. Lediglich wird nun die gewichtete Sobolev-Metrik (5.8) verwendet und die Prototypen werden als Überlagerungen von $G \in \mathbb{N}$ Basisfunktionen $b_i(t)$ dargestellt, so dass gilt:

$$w(t) = \sum_{i=1}^{G} b_i(t).$$

Diese Modifikation des GRLVQ mit Überlagerung von Basisfunktionen wurde für die späteren Tests implementiert und deren Resultate untersucht. Dabei verwendete man als Basisfunktionen die Gauß- und Sigmoidfunktionen.

5.3 Ergebnisse mit realen Daten

Es wurden der GRLVQ und deren Modifikationen wieder an den Datensätzen aus Kapitel 2 getestet. In diesem Abschnitt werden einige Resultate der Tests diskutiert. Dabei verwendet man folgende Bezeichnungen für die Verfahren:

- GRLVQ aus Abschnitt 5.1 (Bez.: GRLVQ)
- GRLVQ mit Sobolev-Metrik aus Abschnitt 5.2 (Bez.: GRLVQ+Sobolev)
- GRLVQ mit Sobolev-Metrik und Überlagerung von Sigmoidfunktionen (Bez.: GLVQ+Sobolev+Sigmoid)

Das Verfahren des GRLVQ mit der Sobolev-Metrik und der zusätzlichen Darstellung der Prototypen durch Gaußfunktionen wird in diesem Abschnitt nicht betrachtet. Bei den Tests des GLVQ für den Tecator-Datensatz im Kapitel 4 wurde gezeigt, dass die Gaußfunktionen zur Darstellung von glatten Prototypen ungeeignet sind. Diese Beobachtung bestätigte sich auch für den GRLVQ.
Die einzelnen Lernraten sind in der folgenden Tabelle (5.1) angegeben.

Verfahren	ε	ε^{β}	ε^{σ}	ε^{θ}	ε^{s}
GRLVQ	0.01	–	–	–	–
GRLVQ+Sobolev	0.01	–	–	–	–
GRLVQ+Sobolev+Sigmoid	–	0.01	0.005	0.005	0.005

Tabelle 5.1: Eingabeparameter für Analysen

Wobei ε für die Lernrate der Prototypen und ε^{β}, ε^{σ}, ε^{θ} und ε^{s} für die Lernrate der Höhe, Breite, Ort und der Verschiebung in y-Richtung der Basisfunktionen stehen. Für die Lernrate des Relevanzprofils $\boldsymbol{\lambda}$ wurde in allen Tests $\varepsilon^{\boldsymbol{\lambda}} = 0.0000005$ und für die Lernrate des Parameters α wurde $\varepsilon^{\alpha} = 0.00005$ genutzt.

Tecator-Datensatz

Es werden zunächst die Resultate des GRLVQ auf den Tecator-Datensatz vorgestellt. Jedes Verfahren wurde mit 8000 Lernzyklen angewendet. In Abbildung 5.2 ist zu erkennen, dass die Prototypen der verschiedenen Modifikationen des GRLVQ sich nicht unterscheiden von denen des GLVQ aus Kapitel 4. Demnach können hierfür die gleichen

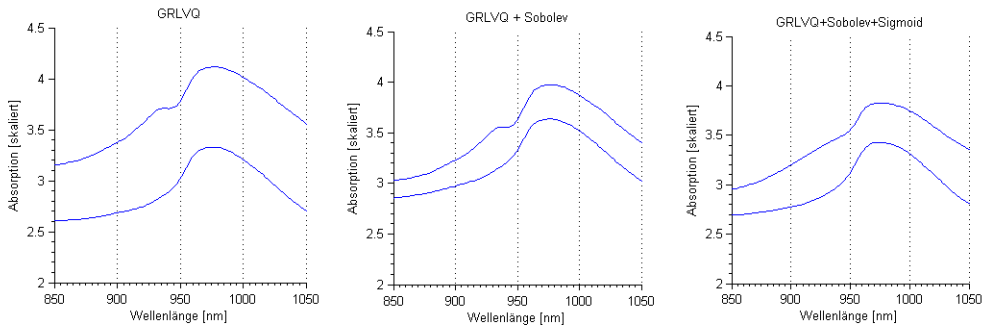

Abbildung 5.2: Prototypen für Tecator-Datensatz

Aussagen getroffen werden, wie bei den Prototypen für die Modifikation des GLVQ. Die Verwendung der Sobolev-Metrik ermöglicht ebenfalls für den GRLVQ eine bessere Lage der Prototypen für den Tecator-Datensatz. Auf Grund der überlappenden Daten legt der GRLVQ die Prototypen weiter auseinander als der 'GRLVQ+Sobolev'. Die Sigmoidfunktionen als Basisfunktionen stellen glatte Prototypen sehr gut dar. Lediglich das lokale Maximum im Dimensionsbereich 40-50 kann durch Sigmoidfunktionen nicht ausgeprägt werden. Außerdem bringt die Verwendung des Relevanzvektors $\boldsymbol{\lambda}$ keine Verbesserung bzw. Änderung hinsichtlich der Lage der Prototypen. In dem späteren Verlauf wird sichtbar, dass dennoch eine zusätzliche Information durch das Relevanzprofil erhalten wird. Ebenfalls die Ergebnisse bezüglich der Genauigkeitsrate r_a der einzelnen Verfahren

sind unverändert im Vergleich mit den Verfahren des GLVQ. Die Genauigkeitsraten der einzelnen Modifikationen des GRLVQ stiegen unter Verwendung der Sobolev-Metrik erheblich an. Sowohl der GRLVQ+Sobolev als auch der GRLVQ+Sobolev+Sigmoid weisen eine 17%-ige bzw. 10%-ige Leistungssteigerung im Vergleich zum GRLVQ auf (siehe Abbildung 5.3). Damit ist gezeigt, dass ebenfalls die Darstellung der Prototypen durch geeignete Basisfunktionen mit dem GRLVQ kombiniert werden kann.
Vergleicht man den GRLVQ+Sobolev mit dem GLVQ+Sobolev aus Kapitel 4 sind keine erheblichen Unterschiede in den Resultaten zu erkennen. Die Prototypen haben ungefähr die gleiche Lage und auch die Genauigkeitsraten sind fast ähnlich. Die zusätzliche Information, die der GRLVQ+Sobolev liefert, ist das Relevanzprofil bzgl. der Wellenlängen. Zunächst betrachtet man die Genauigkeitsraten des GRLVQ und dessen Modifikationen in Abbildung 5.3.

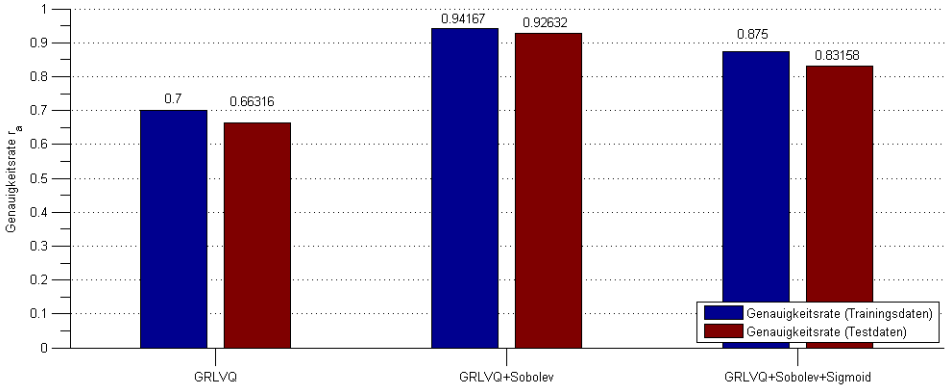

Abbildung 5.3: Genauigkeitsrate der einzelnen Verfahren für den Tecator-Datensatz

Es wird nun ein interessanter Aspekt in diesem Abschnitt betrachtet. Hierfür stellt man sich die Frag: “Welche Auswirkungen hat der Parameter α auf das Relevanzprofil $\boldsymbol{\lambda}$?”
Es wurde sowohl für den vorderen Term der Sobolev-Metrik als auch für den hinteren Ableitungsterm der gleiche Relevanzvektor $\boldsymbol{\lambda}$ verwendet. Zur Erinnerung wird die verwendete Sobolev-Metrik nochmals angegeben:

$$d_{S,\boldsymbol{\lambda}}(\mathbf{x},\mathbf{w},\alpha) = (1-\alpha)\cdot\sum_{t=1}^{n}\left[\lambda(t)\cdot(x(t)-w(t))^2\right] + \alpha\sum_{t=1}^{n}\left[\lambda(t)\cdot(\dot{x}(t)-\dot{w}(t))^2\right]$$

Um die obige Frage beantworten zu können, wird zunächst die Abbildung 5.4 betrachtet.

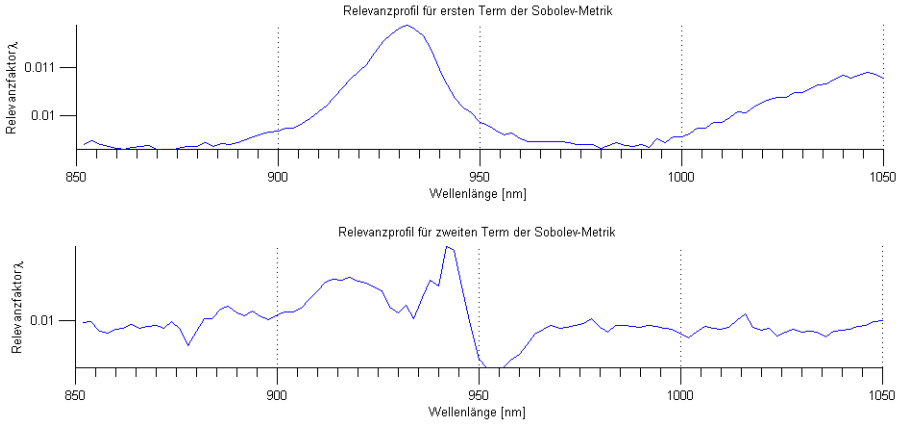

Abbildung 5.4: Relevanzprofil der einzelnen Sobolev-Terme

Für die erste Grafik von oben in Abbildung 5.4 wurde der GRLVQ+Sobolev mit $\alpha = 0$ gesetzt und das Relevanzprofil $\boldsymbol{\lambda}$ lediglich mit der euklidischen Metrik im ersten Term gelernt. An dieser Stelle wird darauf hingewiesen, dass der GRLVQ+Sobolev für $\alpha = 0$ zum klassischen GRLVQ korrespondiert. Für die untere Grafik setzte man $\alpha = 1$ und lernte somit das Relevanzprofil bezüglich der Ableitungen mit dem hinteren Term der Sobolev-Metrik. In der obigen Grafik von Abbildung 5.4 ist Folgendes zu erkennen. Der GRLVQ+Sobolev erlernt eine Relevanzprofil, welches den Bereich zwischen 915nm - 940nm stärker wichtet als andere Wellenlängen. Demnach ist dieser Bereich zur Klassifizierung der Datenvektoren des Tecator-Datensatzes wichtiger als andere Bereiche. Dieses Ergebnis ist jedoch nicht überraschend, da sich die Datenvektoren der unterschiedlichen Klassen genau in diesem Bereich erheblich unterscheiden. Interessanter ist die Betrachtung der unteren Grafik aus Abbildung 5.4. Wenn man die Skalierung der Ordinatenachse berücksichtigt, sind über den gesamten Bereich aller Wellenlängen nur sehr kleine Veränderungen ersichtlich. Man sieht außerdem, dass der GRLVQ+Sobolev für $\alpha = 1$ nur die Ränder des Bereiches 915 - 940 anhebt, jedoch das Innere in einem niedrigen Bereich lässt. Diese Beobachtung lässt sich damit erklären, dass er nun mit dem zweiten Term der Sobolev-Metrik Anstiegsunterschiede betrachtet und keine Distanzen mehr im klassischen Sinne. Er berücksichtigt also nur noch Ähnlichkeiten in den Anstiegen. Im Großen und Ganzen muss man allerdings sagen, dass es keinen Wellenlängenbereich bei den Ableitungen gibt, der durch hohe Relevanz hervorgehoben wird. Außerdem unterscheiden sich die beiden Relevanzprofile aus der Abbildung 5.4 erheblich. Das wiederum legt die Vermutung nahe, einen lokalen Relevanzvektor für den ersten Term der Sobolev-Metrik und einen separaten Relevanzvektor für den zweiten Term zu verwenden. Diese Variante führte allerdings zu keinem zufriedenstellenden Ergebnis. Es konnten zwar die einzelnen Relevanzprofile unabhängig voneinander gelernt werden, der Parameter α jedoch beeinflusste die Lerngeschwindigkeit der Relevenzprofile erheblich. Bei $\alpha = 1$ kam der Lernprozess für das Relevanzprofil des ersten Sobolev-Terms vollständig zum Erliegen. Die Möglichkeit α komplett aus der Metrik zu entfernen, um die Relevanzprofile bestmöglich zu erlernen, bringt bezüglich der Genau-

igkeitsrate schlechte Ergebnisse. Dafür ist der Parameter α zu wichtig für die richtige Klassifizierung.
Für den weiteren Verlauf wurde nun der Parameter α gelernt und dabei die Veränderung des Relevanzprofils untersucht. Hierfür betrachte man die Abbildung 5.5. Diese zeigt den Verlauf des Relevanzprofils in Abhängigkeit des gelernten Parameters α unter der Anwendung des GRLVQ+Sobolev.

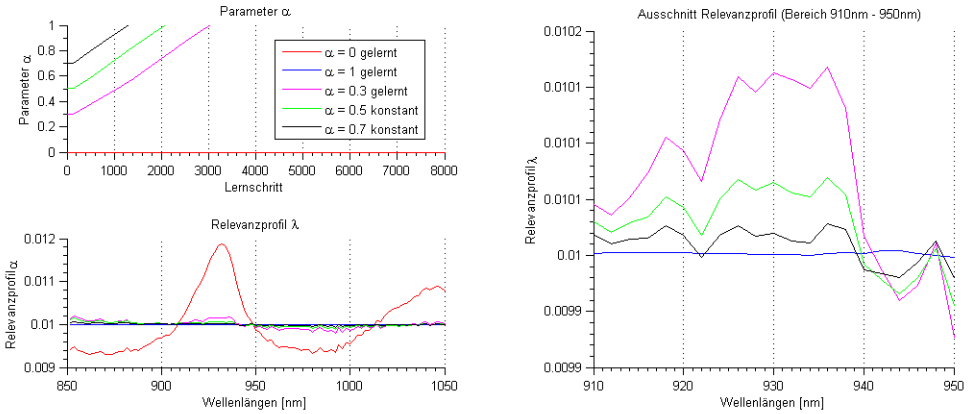

Abbildung 5.5: Relevanzprofil bezüglich des Parameters α

Die obere linke Grafik illustriert den Parameterverlauf von α. Dabei wurden die Startwerte 0.3, 0.5 und 0.7 verwendet. Wie man leicht sieht, steigt der Parameter α unabhängig vom Startwert im Laufe des Lernprozesses auf eins an. Die jeweiligen Relevanzvektoren nach 8000 Lernschritten sind in der unteren linken Grafik dargestellt. Dabei ist zu erkennen, je länger der Parameter $\alpha = 1$ gesetzt war, desto weniger prägt der GRLVQ+Sobolev den Berg im Bereich 915 - 940nm aus. Anfangs versucht der Algorithmus noch diesen Hügel im Relevanzprofil auszubilden, senkt ihn aber dann wieder je näher α an die eins rückt. Die Ursache liegt in den unterschiedlichen Relevanzprofilen für den euklidischen Abstand und für die Ableitungen.
Eine weitere interessante Beobachtung ist in Abbildung 5.6 zu erkennen.

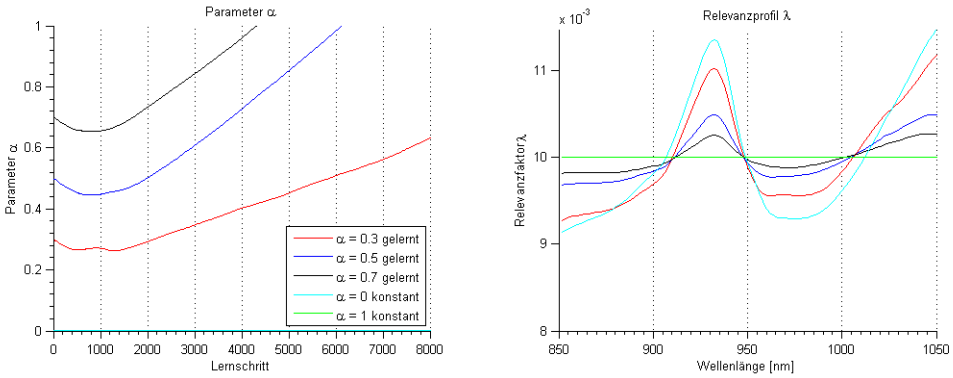

Abbildung 5.6: Relevanzprofil bezüglich des Parameters α (GRLVQ+Sobolev+Sigmoid)

Hier ist das Relevanzprofil des GRLVQ+Sobolev+Sigmoid für verschiedene Startwerte von α dargestellt. Es ist der gleiche Effekt zu erkennen, wie schon in Abbildung 5.5. Je länger bzw. näher der Parameter α bei eins liegt, desto mehr überwiegt der Einfluss des Relevanzprofils der Ableitungen. Nun soll der Fokus auf den Verlauf des Relevanzprofils gerichtet werden. Dieser ist nun glatt im Vergleich zu dem Relevanzprofil des GRLVQ+Sobolev. Das liegt an der Verwendung der Basisfunktionen zur Darstellung der Prototypen. Durch die Adaption der Basisfunktion wird nicht mehr jede Dimension $\lambda(t)$ einzeln betrachtet, sondern es werden noch zusätzlich die Nachbardimensionen adaptiert. Dieses Verfahren kann man in Beziehung setzen mit 'enhancement Generalized Relevance Learning Vectorquantization (eGRLVQ)'. Der eGRLVQ basiert jedoch auf einer anderen Idee. Auf diese soll hier aber nicht näher eingegangen werden. Hierfür wird auf den Vortrag von Marika Kästner zum MiWoCI 2012 verwiesen. Es wurde somit ein anderer Ansatz gefunden, wie man diese zusätzliche Adaption der Nachbarschaftsdimensionen realisieren kann.
Zusammenfassend lässt sich sagen, dass der GRLVQ mit der Sobolev-Metrik modifiziert werden kann. Außerdem ist eine Erweiterung des GRLVQ+Sobolev mit dem Ansatz von Basisfunktionen möglich. All diese Modifikationen erzielen auf dem Tecator-Datensatz erhebliche Leistungssteigerungen. Allerdings ist die Entwicklung des Relevanzprofils sehr instabil, wenn der Parameter α gelernt wird. Dieser Effekt kommt daher, da sich die Relevanzprofile der Distanzen bzw. der Ableitungen stark unterscheiden. Aus diesem Grund wird vorgeschlagen,dass bei dem GRLVQ zunächst der Parameter α zusammen mit den Prototypen unabhängig von dem Relevanzprofil gelernt werden sollte. Dieser gelernte Wert für α wird für den weiteren Lernprozess konstant gesetzt und es kann das Relevanzprofil erneut mit den Prototypen gelernt werden.

Kaffee-Datensatz

Für die Tests des GRLVQ wurde der gleiche Kaffee-Datensatz II wie in Abschnitt 4.3 verwendet. Bereits bei den Tests des GRLVQ auf den Tecator-Datensatz war zu erkennen, dass es keine gravierenden Unterschiede in den Resultaten zum GLVQ gab. Die gleichen Beobachtungen wurden bei den Tests auf dem Kaffee-Datensatz gemacht. Das bedeutet, es gibt keine großen Unterschiede zwischen den Modifikationen des GLVQ und dem GRLVQ auf dem Kaffee-Datensatz. Aus diesem Grund wird auf eine ausführliche Auswertung der Lage der Prototypen und der Genauigkeitsrate verzichtet. Zusammenfassend kann gesagt werden, dass die Sobolev-Metrik für den Kaffee-Datensatz keine Verbesserung des GRLVQ hinsichtlich der Genauigkeitsrate erzielt. Außerdem sind in der Lage der Prototypen kaum Unterschiede im Vergleich zum klassischen GRLVQ zu erkennen.
Der Fokus dieser Tests soll auf die Relevanzprofile gerichtet sein. Hierfür betrachtet man zunächst das Relevanzprofil unter der Beobachtung des Parameters α. In der Abbildung 5.7 sind die Relevanzprofile bezüglich der klassischen Distanzen (erster Sobolev-Term) und bezüglich der Ableitungen (zweiter Sobolev-Term) illustriert. Wie schon bei den

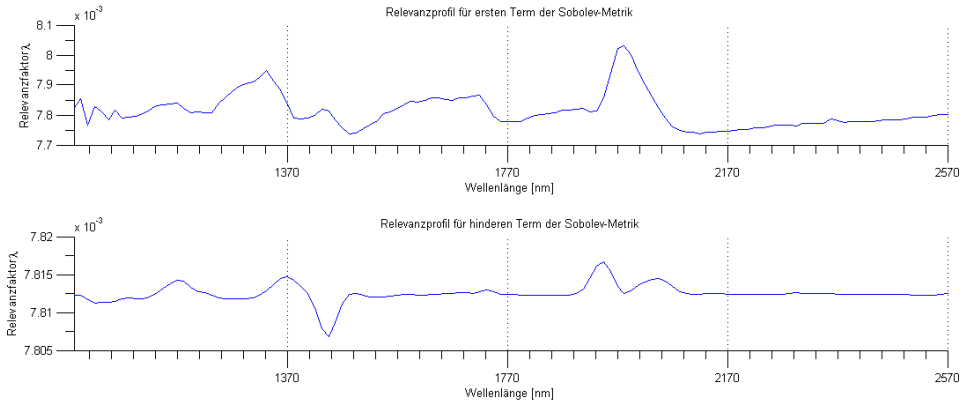

Abbildung 5.7: Relevanzprofil der einzelnen Sobolev-Terme für den GRLVQ+Sobolev

Resultaten für den Tecator-Datensatz ist zu erkennen, dass die klassischen Distanzen und die Ableitungen unterschiedliche Relevanzprofile aufweisen. Wenn der Parameter α zusätzlich gelernt wird, beeinflusst dieser erheblich das Relevanzprofil. Das würde bedeuten, solange der Parameter α nicht konvergiert, konvergiert auch das Relevanzprofil nicht. Man zeigte in den vorangegangenen Tests, dass eine Konvergenz von α nur schwer gewährleistet werden kann. Aus diesem Grund sollte man auch hier zunächst den Parameter α lernen und für den weiteren Lernprozess für das Relevanzprofil konstant setzen.
In Abschnitt 4.3 wurde gezeigt, dass α für den Kaffee-Datensatz gegen 0.4676 konvergierte. Die gleiche Beobachtung wurde bei dem GRLVQ+Sobolev gemacht. In der Abbildung 5.8 ist das Relevanzprofil für $\alpha = 0.4676$ dargestellt. Die Wellenlängenbe-

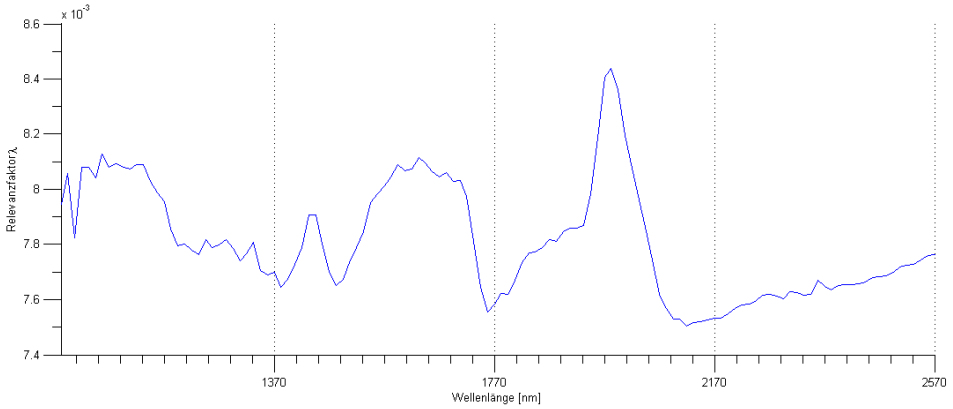

Abbildung 5.8: Relevanzprofil des GRLVQ+Sobolev für $\alpha = 0.4676$

reiche von 1500 - 1700nm und auch bei 1970nm sind laut Relevanzprofil in Abbildung (5.8) sehr wichtig für die Trennung der Datenvektoren. Im Fall des Kaffee-Datensatzes bringt die Erkenntnis über die relevanten Wellenlängen keine entscheidenden Vorteile. In Abbildung (4.8) wurden die Ergebnisse der Verfahren um den GLVQ illustriert. Für den GRLVQ ergaben sich an dieser Stelle keine erheblichen Unterschiede. Es wird somit auf eine Illustration verzichtet.
Zusammenfassend kann man sagen, das Relevenzprofil ist eine zusätzliche Informati-

on über die Struktur der Daten. Dabei ist es sehr schwierig den Parameter α und den Relevanzvektor $\boldsymbol{\lambda}$ gleichzeitig zu lernen. Es wird in dem nächsten Kapitel ein Verfahren vorgestellt, welches die Korrelation zwischen den einzelnen Dimensionen berücksichtigt. Dies ermöglicht eine noch genauere Betrachtung von Klassenunterschieden, die zur Trennung der einzelnen Klassen verwendet werden können.

6 Generalized Matrix Learning Vector Quantization

6.1 Theorie von Generalized Matrix Learning Vector Quantization

'Generalized Matrix Learning Vector Quantization (GMLVQ)' wurde 2008 das erste Mal von Michael Biehl, Petra Schneider und Barbara Hammer in der Publikation [13] veröffentlicht. Dabei wird nach dem gleichen Prinzip vorgegangen, wie bei dem GLVQ aus Abschnitt 4. Man minimiert eine Kostenfunktion durch einen stochastischen Gradientenabstieg, welche die Anzahl der Fehlklassifikationen approximiert.
Sei nun erneut $\mathscr{T} = \{(\mathbf{x},k) \mid \mathbf{x} \in \mathbb{R}^n,\ k \in \mathscr{K}\}$ die Trainingsdatenmenge mit n-dimensionalen Datenvektoren. Anstelle der euklidischen Metrik aus dem GLVQ von Sato und Yamada wird in dem GMLVQ das folgende Ähnlichkeitsmaß [6] verwendet:

$$d_{\mathbf{\Lambda}}(\mathbf{x},\mathbf{w}) = (\mathbf{x}-\mathbf{w})^T \mathbf{\Lambda} (\mathbf{x}-\mathbf{w}), \tag{6.1}$$

wobei $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ eine symmetrische und positiv definite Matrix ist. Diese Matrix $\mathbf{\Lambda}$ gibt eine Korrelation zwischen den Datendimensionen an, die für die Klassifikation wichtig sind. Das Ähnlichkeitsmaß (6.1) ist symmetrisch und positive aber im Allgemeinen kein Distanzmaß.

Bemerkung 6.1 Der Beweis für die Symmetrie und die positive Definitheit des Ähnlichkeitsmaßes (6.1) befindet sich in der Literatur [5].

Nun kann die Matrix $\mathbf{\Lambda}$ zerlegt werden in

$$\mathbf{\Lambda} = \mathbf{\Omega}^T \mathbf{\Omega}, \tag{6.2}$$

wobei $\mathbf{\Omega} \in \mathbb{R}^{m \times n}$ eine reellwertige Matrix ist. Diese Zerlegung (6.2) sichert die Eigenschaft der positiven Definitheit von $\mathbf{\Lambda}$. Dabei kann $m \in \mathbb{N}$ beliebig gewählt werden mit $1 \leq m \leq n$. Es ergibt sich somit für (6.1) die folgende Darstellung:

$$\begin{aligned} d_{\mathbf{\Lambda}}(\mathbf{x},\mathbf{w}) &= (\mathbf{x}-\mathbf{w})^T \mathbf{\Lambda} (\mathbf{x}-\mathbf{w}) \\ &= (\mathbf{x}-\mathbf{w})^T \mathbf{\Omega}^T \mathbf{\Omega} (\mathbf{x}-\mathbf{w}) \\ &= (\mathbf{\Omega}(\mathbf{x}-\mathbf{w}))^T (\mathbf{\Omega}(\mathbf{x}-\mathbf{w})) \\ &= (\mathbf{\Omega}(\mathbf{x}-\mathbf{w}))^2 \end{aligned} \tag{6.3}$$

[6] Der Begriff Ähnlichkeits- bzw. Unähnlichkeitsmaß wurde das erste Mal 2005 von Duin und Pekalska in [8] eingeführt.

Es wird nochmal darauf hingewiesen, dass diese Metrik im Allgemeinen kein Distanzmaß ist, sondern ein Ähnlichkeitsmaß. Für den GMLVQ ergibt sich nun die Kostenfunktion

$$E^{GMLVQ} = \sum_{\mathbf{x}\in\mathscr{T}} f(\mu_{\mathbf{\Lambda}}(\mathbf{x})), \tag{6.4}$$

wobei

$$\mu_{\mathbf{\Lambda}}(\mathbf{x}) = \frac{d^+_{\mathbf{\Lambda}}(\mathbf{x}) - d^-_{\mathbf{\Lambda}}(\mathbf{x})}{d^+_{\mathbf{\Lambda}}(\mathbf{x}) + d^-_{\mathbf{\Lambda}}(\mathbf{x})}.$$

Die Größen $d^+_{\mathbf{\Lambda}}(\mathbf{x})$ bzw. $d^-_{\mathbf{\Lambda}}(\mathbf{x})$ stellen wieder die Distanzen des Gewinner- bzw. Verliererprototypen zu dem Datenvektor $\mathbf{x}$ dar. Wie schon bei dem GLVQ wird also in jedem Lernschritt ein Datenvektor zufällig ausgewählt und dessen Gewinner- und Verliererprototyp bestimmt. Diese werden nun durch einen stochastischen Gradientenabstieg der Kostenfunktion (6.4) adaptiert:

$$\mathbf{w}^{\pm} \leftarrow \mathbf{w}^{\pm} - \varepsilon_r \frac{\partial E^{GMLVQ}}{\partial \mathbf{w}^{\pm}}, \tag{6.5}$$

wobei

$$\frac{\partial E^{GMLVQ}}{\partial \mathbf{w}^{\pm}} = \frac{\partial f}{\partial \mu_{\mathbf{\Lambda}}} \cdot \frac{\partial \mu_{\mathbf{\Lambda}}}{\partial d^{\pm}_{\mathbf{\Lambda}}(\mathbf{x})} \cdot \frac{\partial d^{\pm}_{\mathbf{\Lambda}}(\mathbf{x})}{\partial \mathbf{w}^{\pm}} \tag{6.6}$$

und

$$\frac{\partial d^{\pm}_{\mathbf{\Lambda}}(\mathbf{x})}{\partial \mathbf{w}^{\pm}} = -2\mathbf{\Lambda}(\mathbf{x} - \mathbf{w}^{\pm})$$

Es wird nun zusätzlich in jedem Lernschritt r die Matrix $\mathbf{\Omega}$ durch eine stochastischen Gradientenabstieg adaptiert:

$$\mathbf{\Omega} \leftarrow \mathbf{\Omega} - \varepsilon^{\mathbf{\Omega}}_r \frac{\partial E^{GMLVQ}}{\partial \mathbf{\Omega}}, \tag{6.7}$$

wobei

$$\frac{\partial E^{GMLVQ}}{\partial \mathbf{\Omega}} = \frac{\partial f}{\partial \mu_{\mathbf{\Lambda}}} \left(\frac{\partial \mu_{\mathbf{\Lambda}}}{\partial d^+_{\mathbf{\Lambda}}(\mathbf{x})} \cdot \frac{\partial d^+_{\mathbf{\Lambda}}(\mathbf{x})}{\partial \mathbf{\Omega}} + \frac{\partial \mu_{\mathbf{\Lambda}}}{\partial d^-_{\mathbf{\Lambda}}(\mathbf{x})} \cdot \frac{\partial d^-_{\mathbf{\Lambda}}(\mathbf{x})}{\partial \mathbf{\Omega}} \right)$$

und

$$\frac{\partial d^+_{\mathbf{\Lambda}}}{\partial \mathbf{\Omega}} = 2\mathbf{\Omega}(\mathbf{x} - \mathbf{w}^+)(\mathbf{x} - \mathbf{w}^+)^T$$
$$\frac{\partial d^-_{\mathbf{\Lambda}}}{\partial \mathbf{\Omega}} = 2\mathbf{\Omega}(\mathbf{x} - \mathbf{w}^-)(\mathbf{x} - \mathbf{w}^-)^T$$

Die Metrik muss wieder langsamer gelernt werden als die Prototypen (siehe Bemerkung 4.7). Daher gilt erneut:
$\varepsilon^{\mathbf{\Omega}}_r << \varepsilon_r, \ \forall r$. Anders als bei der Matrix $\mathbf{\Lambda}$ ist es nun sehr schwierig, die Einträge der Matrix $\mathbf{\Omega}$ zu interpretieren. Man kann lediglich sagen, dass diese Matrix eine lineare Abbildung $\mathbf{\Omega} : \mathbb{R}^n \to \mathbb{R}^m$ ist. Für eine Visualisierung der Klassenaufteilung sollte $m = 2$ oder $m = 3$ gewählt werden. Das bedeutet, der GMLVQ lernt somit sukzessive eine

lineare Abbildung der Datenvektoren in den Raum $\mathbb{R}^m$, in dem die Daten besser getrennt werden können. Die Matrix $\mathbf{\Omega}$ wird nach der Adaption mit der Formel

$$\Omega_{(i,j)} = \frac{\Omega_{(i,j)}}{\sqrt{\sum_{k=1}^{n} \Lambda_{(k,k)}}}, \tag{6.8}$$

normiert, wobei

$$\Lambda_{(i,i)} = \sum_{j=1}^{m} \left(\Omega_{(j,i)}\right)^2.$$

Damit erreicht man eine höhere Stabilität des Verfahrens und eine bessere Interpretation der Korrelationen zwischen einzelnen Dimensionen untereinander. Die gelernten Prototypen haben somit nach dem Lernprozess die Eigenschaft, dass sie in vielen Fällen eine sehr gute Genauigkeitsrate bzgl. der Datensätze aufweisen, allerdings nicht mehr datentypisch aussehen müssen. Um wieder eine ähnliche Darstellung von Prototypen und Datenvektoren zu erreichen, können Datenvektoren und Prototypen in den Raum $\mathbb{R}^m$ transformiert werden mit

$$\widehat{\mathbf{x}} = \mathbf{\Omega}\mathbf{x}, \ \forall \mathbf{x} \in \mathscr{X} \tag{6.9}$$

$$\widehat{\mathbf{w}} = \mathbf{\Omega}\mathbf{w}, \ \forall \mathbf{w} \in W, \tag{6.10}$$

wobei $\mathbf{\Omega} \in \mathbb{R}^{m \times n}$ und $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$ sind.
Der GMLVQ ist nach dem GLVQ ein weiterer Nachfolger des LVQ und stellt durch das Matrix-Lernen ein äußerst mächtiges Werkzeug dar. Im Folgenden sind die wichtigsten Schritte des Algorithmus nochmal zusammengefasst.

Algorithm 7 GMLVQ

Eingabe: Lernraten ε und $\varepsilon^{\mathbf{\Omega}}$, Trainingsdatensatz $\mathscr{T}$, maximale Anzahl der Iterationen r_{max} und Anzahl der Prototypen pro Klasse
Ausgabe: Prototypen und Matrix $\mathbf{\Lambda}$
zufällige Initialisierung der Prototypen $\mathbf{w}$
Initialisierung von $\mathbf{\Omega} = \mathbb{E}$
Setze r:=0
while $r < r_{max}$ **do**
  Wähle ein $\mathbf{x} \in \mathscr{T}$ zufällig
  Ermittle Abstand zwischen allen Prototypen $\mathbf{w}$ und dem Datenpunkt $\mathbf{x}$ mit (6.3)
  Ermittle Gewinner- und Verliererprototyp von Datenpunkt $\mathbf{x}$
  Adaptiere Gewinner- und Verliererprototyp nach (6.5)
  Adaptiere Matrix $\mathbf{\Omega}$ mit (6.7)
  Normiere Matrix $\mathbf{\Omega}$ mit (6.8)
  $r = r + 1$
end while
Berechne Matrix $\mathbf{\Lambda}$ mit (6.2)

Bemerkung 6.2 Für den Fall, dass $m < n$ gewählt wird, kann $\mathbf{\Omega}$ auch zufällig initialisiert werden. Des Weiteren können die Prototypen mit anderen Verfahren (z.B. GLVQ) vorgelernt werden. Somit sind die Prototypen zu Beginn des Lernprozesses mit dem GMLVQ in einer ähnlichen Form. Dies bewirkt eine Verkürzung des Lernprozesses des GMLVQ.

6.2 Modifikation des GMLVQ

In diesem Abschnitt soll der GMLVQ dahingehend abgewandelt werden, dass nun die Prototypen durch Überlagerungen von Basisfunktionen dargestellt werden. Eine zusätzliche Modifikation des GMLVQ durch die Sobolev-Metrik, wie bei dem GLVQ und GRLVQ aus den vorherigen Abschnitten, ist an dieser Stelle überflüssig. Auf Grund der Matrix $\mathbf{\Lambda}$ ist eine zusätzliche Nachbarschaftskorrelation der Datenvektoren im GMLVQ bereits vorhanden. Das bedeutet, die Anstiege der Datenvektoren werden durch das Matrix-Lernen bereits berücksichtigt.
Sei erneut $\mathscr{T} = \{(\mathbf{x},k) \mid \mathbf{x} \in \mathbb{R}^n,\ k \in \mathscr{K}\}$ eine Trainingsdatenmenge mit funktionalen Daten. Das bedeutet, ein Datenvektor wird als Folge von Funktionswerten einer Funktion interpretiert. Somit gilt wie schon in den vorherigen Abschnitten $\mathscr{T} \subset \mathscr{X} \subseteq C^1[1,n]$. Es wird ein Basissystem $\mathscr{B} = \{b_i(t) \mid i \in \mathscr{I}\}$ eingeführt, wobei $\mathscr{I}$ eine endliche Indexmenge ist. Hierfür gilt erneut die Bemerkung 4.8. Die Prototypen werden durch eine Überlagerung von Basisfunktionen (4.21) dargestellt. Es werden in diesem Abschnitt die Gaußfunktion und Sigmoidfunktion als Basisfunktion verwendet.

Gaußfunktionen als Basissystem

Es werden in diesem Abschnitt die Überlagerung von Gaußfunktionen (4.20) betrachtet. Im folgenden Verlauf wird nicht näher auf die Darstellung der Prototypen eingegangen. Hierfür wird auf das Kapitel 4 verwiesen. Vielmehr ändert sich nun die Metrik (6.3) zu

$$\begin{aligned} d_{\mathbf{\Lambda}}(\mathbf{x},\mathbf{w}) &= (\mathbf{\Omega}(\mathbf{x}-\mathbf{w}))^2 \\ &= \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Omega_{(i,t)} \left(x(t) - w(t)\right) \right)^2 \\ &= \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Omega_{(i,t)} \left(x(t) - \sum_{i=1}^{G} \frac{\beta_i}{\sqrt{2\pi}\cdot\sigma_i} e^{-\frac{(t-\theta_i)^2}{2\cdot\sigma_i^2}} \right) \right)^2 . \end{aligned} \tag{6.11}$$

Es wird nun die Kostenfunktion (6.4) durch einen stochastischen Gradientenabstieg minimiert, wobei nun die relative Fehlerdistanz $\mu_{\mathbf{\Lambda}}(\mathbf{x})$ mit der Metrik (6.11) dargestellt wird. Zur Adaption der Prototypen werden die Höhen β_i und Orte θ_i der Gaußfunktionen ge-

lernt, so dass sich folgende Update-Regeln ergeben:

$$\beta_i^{\pm} \leftarrow \beta_i^{\pm} - \varepsilon_r^{\beta} \frac{\partial E^{GMLVQ}}{\partial \beta_i^{\pm}}$$
$$\theta_i^{\pm} \leftarrow \theta_i^{\pm} - \varepsilon_r^{\theta} \frac{\partial E^{GMLVQ}}{\partial \theta_i^{\pm}}$$

Dabei initialisiert man die Breiten σ_i und hält diese über den gesamten Lernprozess konstant (siehe Bemerkung 4.9). Für die Gradienten der Kostenfunktion ändert sich in (6.6) lediglich $\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \mathbf{w}^{\pm}}$ zu dem Term $\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \beta_i^{\pm}}$ bzw. $\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \theta_i^{\pm}}$. Aus diesem Grund werden an dieser Stelle nur die geänderten Terme angegeben:

$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \beta_i^{\pm}} = -\frac{1}{\sqrt{2\pi}\sigma_i^{\pm}} \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Lambda_{(j,t)} \left(x(t) - w^{\pm}(t) \right) e^{-\frac{(t-\theta_i^{\pm})^2}{2\sigma_i^{\pm 2}}} \right)$$
$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \theta_i^{\pm}} = -\frac{\beta_i^{\pm}}{\sqrt{2\pi}\sigma_i^{\pm 3}} \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Lambda_{(j,t)} \left(x(t) - w^{\pm}(t) \right) \left(t - \theta_i^{\pm} \right) e^{-\frac{(t-\theta_i^{\pm})^2}{2\sigma_i^{\pm 2}}} \right)$$

Auf eine ausführliche Darstellung des Algorithmus wird an dieser Stelle verzichtet. Es ändert sich lediglich die Zeile mit der Adaption der Prototypen in dem Algorithmus 7. Hierfür wird die Adaption von Ort und Höhe der jeweiligen Gaußfunktionen eingesetzt. Die Update-Regel der Matrix $\mathbf{\Omega}$ ist analog zum klassischen GMLVQ aus Abschnitt 6.1.

Sigmoidfunktionen als Basissystem

In diesem Abschnitt wird die Überlagerung von Sigmoidfunktionen (4.28) betrachtet. Wie bereits in Abschnitt 4.2.2 angesprochen, erreicht man durch die Anwendung von Sigmoidfunktionen eine höhere Flexibilität bei der Darstellung der Prototypen im Vergleich zu den Gaußfunktionen.
Es ändert sich an dieser Stelle erneut die Metrik (6.3) zu

$$d_{\mathbf{\Lambda}}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Omega_{(i,t)} \left(x(t) - \sum_{i=1}^{G} \left(\frac{\beta_i}{1 + e^{-\frac{(t-\theta_i)}{2\sigma_i^2}}} + s_i \right) \right) \right)^2 . \tag{6.12}$$

Minimiert man die Kostenfunktion (6.4) durch einen stochastischen Gradientenabstieg, ergeben sich die folgenden Update-Regeln für $\beta_i^{\pm}$, $\sigma_i^{\pm}$, $\theta_i^{\pm}$ und $s_i^{\pm}$:

$$\beta_i^{\pm} \leftarrow \beta_i^{\pm} - \varepsilon_r^{\beta} \frac{\partial E^{GMLVQ}}{\partial \beta_i^{\pm}}$$
$$\theta_i^{\pm} \leftarrow \theta_i^{\pm} - \varepsilon_r^{\theta} \frac{\partial E^{GMLVQ}}{\partial \theta_i^{\pm}}$$

$$\sigma_i^{\pm} \leftarrow \sigma_i^{\pm} - \varepsilon_r^{\sigma} \frac{\partial E^{GMLVQ}}{\partial \sigma_i^{\pm}}$$
$$s_i^{\pm} \leftarrow s_i^{\pm} - \varepsilon_r^{s} \frac{\partial E^{GMLVQ}}{\partial s_i^{\pm}}$$

Es wird somit jeder Parameter in der Sigmoidfunktion gelernt. Wie bereits bei den Gaußfunktionen werden für die Gradienten nur die Teilformeln angegeben, die sich im Wesentlichen ändern:

$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \beta_i^{\pm}} = -\sum_{j=1}^{m} \left(\sum_{t=1}^{n} \frac{\Lambda_{(j,t)} \left(x(t) - w^{\pm}(t)\right)}{\left(1 + e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}\right)} \right)$$

$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \sigma_i^{\pm}} = \frac{\beta_i^{\pm}}{\sigma_i^{\pm 3}} \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \frac{\Lambda_{(j,t)} \left(x(t) - w^{\pm}(t)\right) (t - \theta_i^{\pm}) e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}}{\left(1 + e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}\right)^2} \right)$$

$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial \theta_i^{\pm}} = \frac{\beta_i^{\pm}}{\sigma_i^{\pm 2}} \sum_{j=1}^{m} \left(\sum_{t=1}^{n} \frac{\Lambda_{(j,t)} \left(x(t) - w^{\pm}(t)\right) e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}}{\left(1 + e^{-\frac{(t-\theta_i^{\pm})}{2\sigma_i^{\pm 2}}}\right)^2} \right)$$

$$\frac{\partial d_{\mathbf{\Lambda}}^{\pm}(\mathbf{x})}{\partial s_i^{\pm}} = -\sum_{j=1}^{m} \left(\sum_{t=1}^{n} \Lambda_{(j,t)} \left(x(t) - w^{\pm}(t)\right) \right)$$

Auch hier wird aus dem gleichen Grund, wie bei den Gaußfunktionen auf eine Darstellung des modifizierten Algorithmus 7 verzichtet.

6.3 Ergebnisse mit realen Daten

Der GMLVQ mit seinen Modifikationen wurde nun an den realen Daten aus Kapitel 2 getestet. Dabei werden die Verfahren wieder wie folgt bezeichnet:

- GMLVQ aus Abschnitt 6.1 (Bez.: GMLVQ)
- GMLVQ mit Überlagerung von Gaußfunktionen aus Abschnitt 6.2 (Bez.: GMLVQ+Gauß)
- GMLVQ mit Überlagerung von Sigmoidfunktionen aus Abschnitt 6.2

(Bez.: GMLVQ+Sigmoid)

Aufgrund der langsamen Ausprägung der Matrix $\mathbf{\Lambda}$ wurden für jeden Test die Verfahren mit 10000 Lernzyklen gelernt. Für die Approximation der relativen Fehlerdistanz wurde wieder die Identitätsfunktion verwendet. Des Weiteren initialisierte man einen Prototyp pro Klasse und lernte diese mit dem GLVQ vor. Es wurden für die Lernraten in allen Tests folgende Werte verwendet.

Verfahren	ε	ε^{β}	ε^{σ}	ε^{θ}	ε^{s}
GMLVQ	0.01	–	–	–	–
GMLVQ+Gauß	–	0.01	–	0.005	–
GMLVQ+Sigmoid	–	0.01	0.005	0.005	0.005

Tabelle 6.1: Eingabeparameter für Analysen

Des Weiteren wurde die Matrix $\mathbf{\Omega}$ mit der Lernrate $\varepsilon^{\mathbf{\Omega}} = 0.0001$ gelernt und stets als (m,n)-Einheitsmatrix initialisiert ($m = n$).
Für alle Tests steht vor allem die Entwicklung und Ausprägung der Matrix $\mathbf{\Lambda}$ im Fokus. Außerdem wird die Genauigkeitsrate der einzelnen Verfahren begutachtet. Eine Betrachtung der Prototypen ist an dieser Stelle ungünstig, da deren Form nicht mehr mit dem Verlauf der Daten übereinstimmen muss. Daher werden die abgebildeten Prototypen und Datenvektoren mit den Formeln (6.10) und (6.9) betrachtet.

Tecator-Datensatz

Es wurden für den Tecator-Datensatz nur die Verfahren GMLVQ und GMLVQ+Sigmoid getestet. Alle vorherigen Tests mit Verfahren und Überlagerungen von Gaußfunktionen ergaben auf dem Tecator-Datensatz keine zufriedenstellende Ergebnisse. Aus diesem Grund wurde der GMLVQ+Gauß nur auf dem Kaffee-Datensatz getestet. Des Weiteren wurden 10 Sigmoidfunktionen als Basisfunktionen zur Darstellung der Prototypen in dem GMLVQ+Sigmoid verwendet.
Man betrachtet zunächst die Genauigkeitsraten der einzelnen Verfahren GMLVQ und GMLVQ+Sigmoid in Abbildung 6.1. Beide Verfahren erzielen verschiedene Resultate. Der GMLVQ+Sigmoid verliert durch die Anwendung von Sigmoidfunktionen zur Darstellung der Prototypen ein wenig an Genauigkeit. Vermutlich können die Prototypen nicht so detailliert dargestellt werden, um eine ähnliche Performance wie der GMLVQ zu erreichen. Jedoch ist für den Tecator-Datensatz gezeigt, dass die Darstellung von Prototypen durch Sigmoidfunktionen im GMLVQ möglich ist.
Man betrachte nun die Abbildung 6.2.

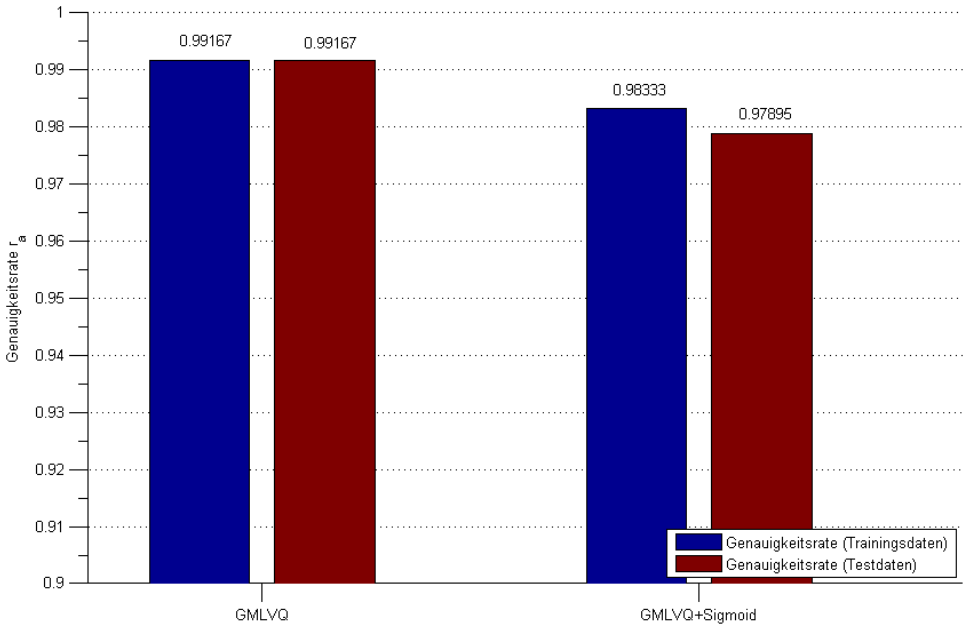

Abbildung 6.1: Genauigkeitsrate der einzelnen Verfahren für den Tecator-Datensatz

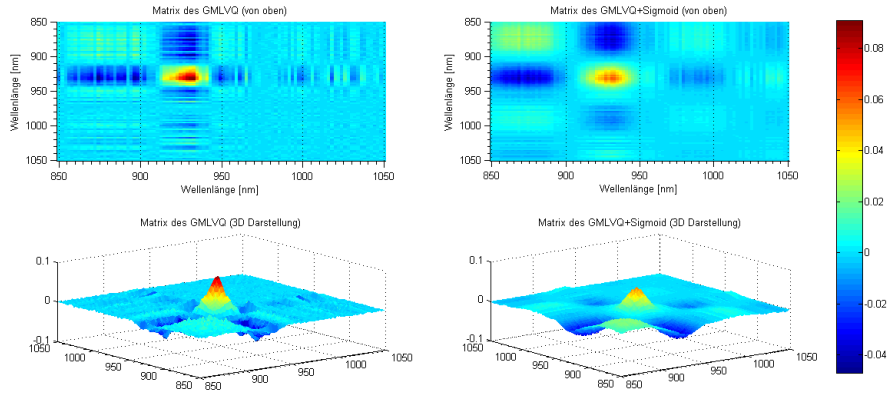

Abbildung 6.2: Ausprägung der Matrix $\mathbf{\Lambda}$ von oben und in 3D-Darstellung

Zunächst wird auf die große Korrelation der Datenvektoren in dem Wellenlängenbereich 910nm bis 940nm hingewiesen. Beide Verfahren prägen bei dem Lernen der Matrix $\mathbf{\Lambda}$ diese Bereiche sehr gut aus. Das ist ein sehr wichtiger Grund, wieso diese beiden Verfahren auf dem Tecator-Datensatz so hervorragende Ergebnisse erzielen.
Ein weiterer interessanter Effekt ist bei der Ausprägung der Matrix $\mathbf{\Lambda}$ zu beobachten. Wie man in den unteren beiden 3D-Darstellungen der Matrizen $\mathbf{\Lambda}$ erkennen kann, wurde durch die Anwendung der Sigmoidfunktion das Relevanzprofil bzw. die Matrix $\mathbf{\Lambda}$ geglättet. Diese Beobachtung ist bereits bei dem GRLVQ aus Abschnitt 5.3 aufgetreten. Aufgrund der Basisdarstellung der Prototypen werden nicht mehr die einzelnen Dimensionen der Prototypen adaptiert, sondern lediglich deren Basisfunktionen. Somit entsteht gezwungenermaßen eine Nachbarschaftsbeziehung zwischen benachbarten

Dimensionen. Diese Nachbarschaftsbeziehung wirkt sich auch auf die Relevanzen der einzelnen Dimensionen aus. Das bedeutet: Wenn bestimmte Dimension mit hoher Relevanz gelernt werden, so werden automatisch auch deren Nachbarn mit erhöhter Relevanz gelernt. Das Resultat ist ein glattes Relevanzprofil. Dieser Effekt ist einerseits ein großer Vorteil, weil Rauschen bzw. irrelevante Informationen nicht mit berücksichtigt werden. Es kann sich allerdings zum Nachteil entwickeln, wenn in dem Rauschen diverse Informationen bezüglich der Klassentrennung enthalten sind. Bei einer genaueren Beobachtung des Hügels in Abbildung 6.2, sind beispielsweise unterschiedliche Höhen zu erkennen. Der Gipfel des GMLVQ+Sigmoid ist kleiner ausgeprägt, da benachbarte Dimensionen (z.B. am Fuß des Maximums) mit einer kleineren Relevanz der Ausprägung des Gipfels entgegenwirkten. Somit erhält man eine kleinere Korrelation als es bei dem Verfahren GMLVQ ist. Dieser Umstand könnte auch ein Grund dafür sein, wieso der GMLVQ+Sigmoid ein wenig schlechter in der Genauigkeitsrate ist als der GMLVQ. Zusammenfassend lässt sich sagen, dass beide Verfahren sehr gute Ergebnisse auf dem Tecator-Datensatz erzielen.

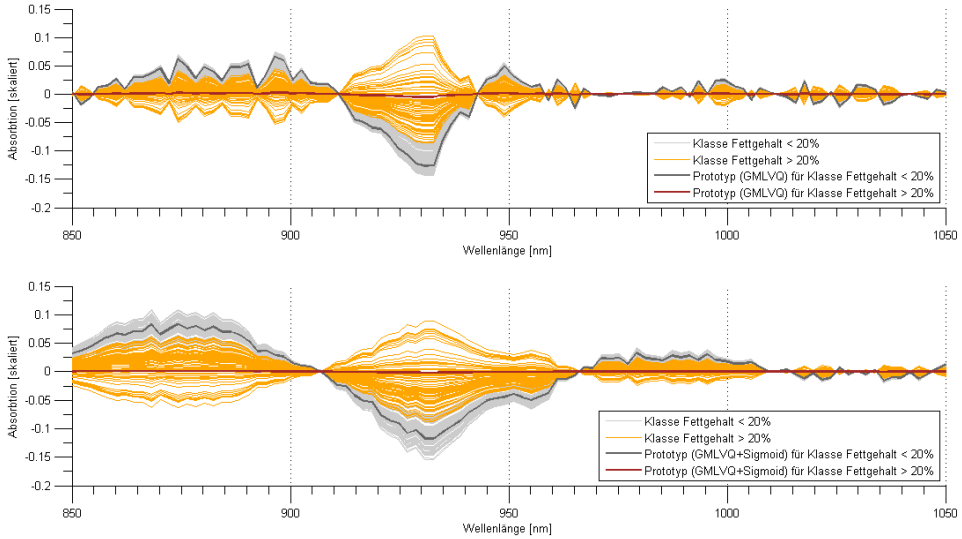

Abbildung 6.3: oben: transformierter Tecator-Datensatz nach GMLVQ / unten: transformierter Tecator-Datensatz nach GMLVQ+Sigmoid

Abschließend sind die transformierten Datenvektoren und Prototypen des Tecator-Datensatzes nach den Formeln (6.9) und (6.10) in Abbildung 6.3 illustriert. Es wird deutlich, dass die transformierten Datenvektoren keine Ähnlichkeit mehr zu den Originaldaten haben. In dem Bildraum sind jedoch die Klassen der Datenvektoren einfacher trennbar als in dem Urbildraum. Daher auch die sehr gute Performanz des GMLVQ bzw. GMLVQ+Sigmoid.

Kaffee-Datensatz

Es handelt sich um den Kaffee-Datensatz II aus Abschnitt 2.2. An diesem Datensatz testete man die Verfahren GMLVQ, GMLVQ+Gauß und GMLVQ+Sigmoid. Für den GMLVQ+Gauß wurden 20 Gaußfunktionen pro Prototyp verwendet und für den GMLVQ+Sigmoid 32 Sigmoidfunktionen. Die Anzahl der Basisfunktionen wurde bei beiden Verfahren im Vergleich zum Tecator-Datensatz etwas höher gewählt, weil die Datenvektoren sehr wellige Verläufe aufweisen.
Man betrachtet zunächst wieder die Genauigkeitsraten der einzelnen Verfahren, um die Performance untereinander zu vergleichen. In der Abbildung 6.4 sind die Genauigkeits-

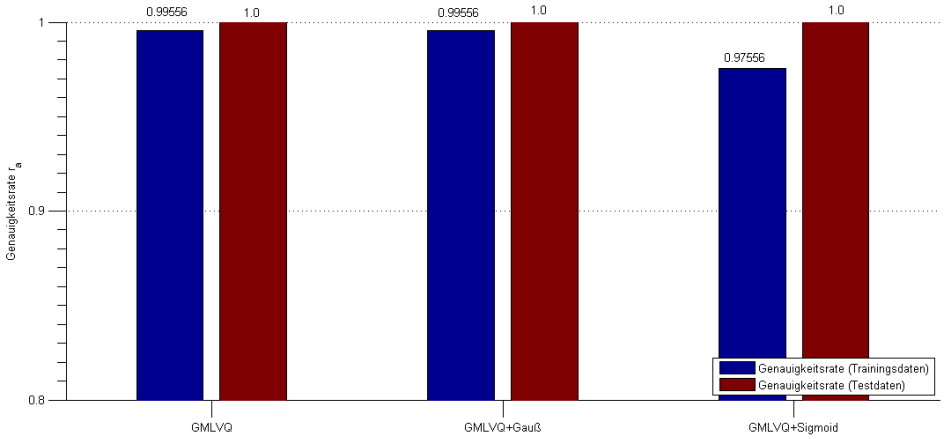

Abbildung 6.4: Genauigkeitsrate der einzelnen Verfahren für den Kaffee-Datensatz

raten der besten Ergebnisse illustriert. Dabei ist zu erkennen, dass alle Verfahren auf dem Testdatensatz alle Datenvektoren zu 100% richtig klassifizieren. Außerdem erreichen der GMLVQ und der GMLVQ+Gauß auf dem Trainingsdatensatz ähnlich gute Ergebnisse von rund 99.6%. Lediglich der GMLVQ+Sigmoid erzielt ein wenig schlechtere Ergebnisse mit 97.6%. Das kann an der ungünstigen Wahl der Basisfunktionen liegen bzw. an dem ungünstigen Ablauf des Lernprozesses. Aber im Allgemeinen sind diese Ergebnisse sehr gut und das ist ein weiteres Argument dafür, dass die Prototypen im GMLVQ als Überlagerung von Basisfunktionen für den Kaffee-Datensatz dargestellt werden können, ohne dass sich die Performanz verschlechtert.
Betrachtet man nun die Matrix $\mathbf{\Lambda}$ jeweils von den drei Verfahren in Abbildung 6.5, ist ebenfalls der Effekt eines glatten Relevanzprofils zu erkennen, wie schon bei dem Tecator-Datensatz. Die Verwendung von Basisfunktionen bewirkt also auch hier eine Nachbarschaftskorrelation von benachbarten Dimensionen. Das wurde bereits in dem Test des GMLVQ zum Tecator-Datensatz diskutiert. Es wird an dieser Stelle nochmal nachdrücklich darauf hingewiesen, dass diese Korrelation zwischen benachbarten Dimensionen kritisch betrachtet werden sollte. Die Diagonalen in den Grafiken aus Abbildung 6.5 erscheint aus dem Grund, weil die Matrix $\mathbf{\Omega}$ als Einheitsmatrix initialisiert wurde. Das ist ein Indiz dafür, dass die Matrix nach 10000 Lernschritten noch nicht endgültig fertig gelernt war. Des Weiteren ist eine hohe positive Korrelation zwischen

den Wellenlängenbereichen um 2000nm zu erkennen. In der obersten linken Abbildung von (6.5) ist die Korrelationsmatrix $\mathbf{\Lambda}$ des GMLVQ illustriert. Die stark positiv korrelierte Wellenlängen sind im Vergleich zu denen vom Verfahren GMLVQ+Gauß relativ gut ausgeprägt. Der GMLVQ+Gauß glättet diese kleinen Korrelationen durch die Verwendung von Basisfunktionen weg. Daher werden in der mittleren linken Abbildung von (6.5) die positiven Korrelationen nicht so stark ausgebildet, bis auf die Umgebung von Wellenlänge 2000nm.
Zusammenfassend lässt sich sagen, dass alle Verfahren GMLVQ, GMLVQ+Gauß und GMLVQ+Sigmoid sehr gute Ergebnisse auf dem reduzierten Datensatz erzielen. Einziger Unterschied ist die Darstellung der Korrelationsmatrix bei dem GMLVQ+Basisfunktionen. In dieser werden kaum positive Korrelationen ersichtlich, auf Grund der Glättung kleiner Veränderungen. Dennoch erreichen diese Verfahren sehr gute Ergebnisse.

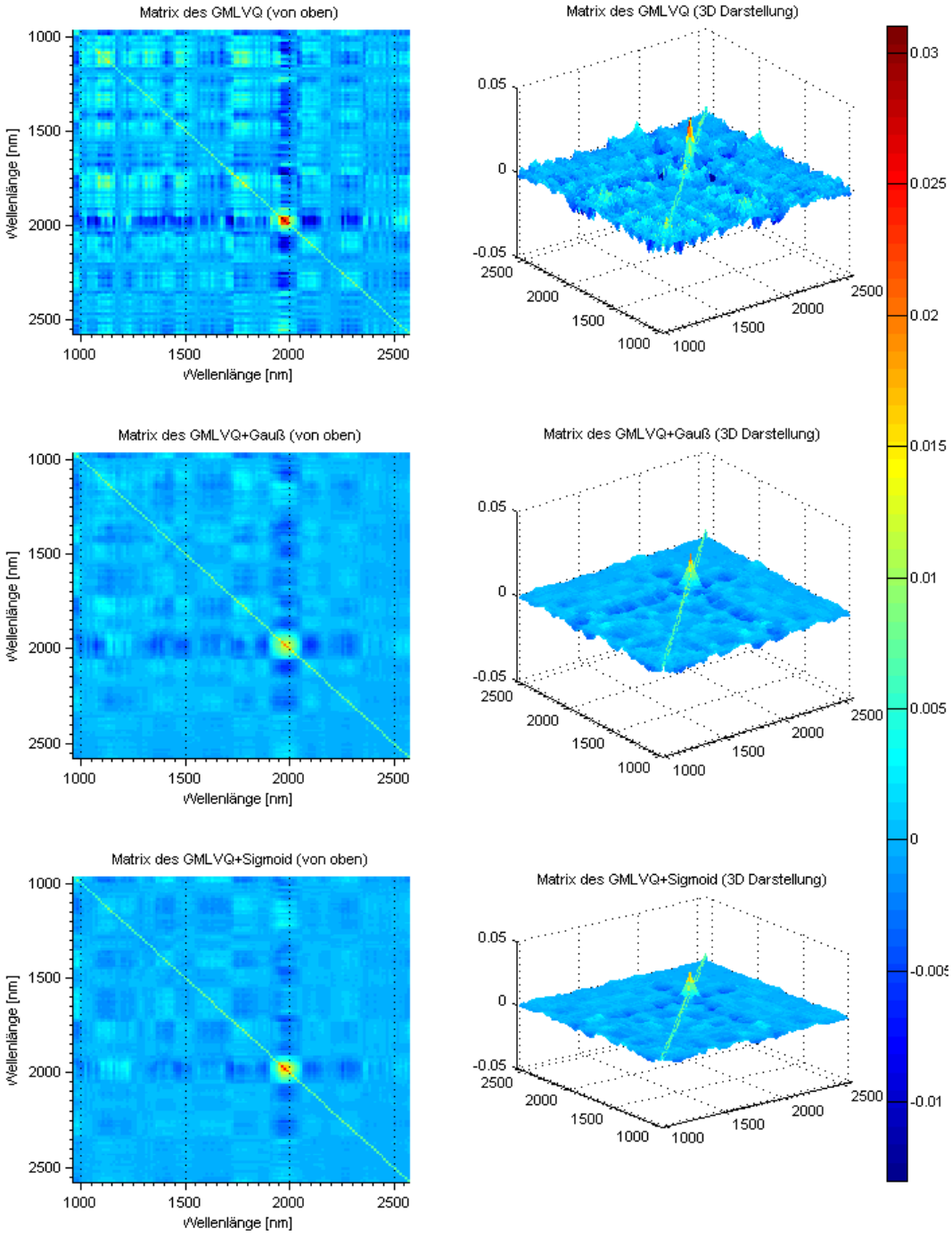

Abbildung 6.5: Ausprägung der Matrix $\mathbf{\Lambda}$ von oben und in 3D-Darstellung

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Teilmenge von Verfahren des überwachten Lernens auf funktionale Daten angepasst. Als Verfahren betrachtete man den GLVQ, GRLVQ und GMLVQ. Bei der Anpassung dieser Methoden wurden die funktionalen Eigenschaften der Datenvektoren so genutzt, dass möglichst gute Ergebnisse bei der Klassifikation der Daten erzielt wurden.
Es wurde zuerst gezeigt, dass die vereinfachte Sobolev-Norm äquivalent zur originalen Sobolev-Norm ist. Nun ersetzte man in den Verfahren GLVQ und GRLVQ die euklidische Metrik durch die vereinfachte Sobolev-Metrik (4.23) bzw. (5.8). Damit betrachtete man zu dem euklidischen Abstand der Datenvektoren und Prototypen zusätzlich die Abstände der Anstiege. Außerdem wurde der Abstand zwischen den Anstiegen mit einem Parameter $\alpha \in [0,1]$ gewichtet. Die Parameter wurde ebenfalls durch einen stochastischen Gradientenabstieg adaptiert. Das hat den Vorteil, dass das Verfahren flexibler auf verschiedene Datensätze angewendet werden kann. Sollten die Anstiege der Daten zur Klassifikation wichtig sein, dann wird der Parameter α größer. Dieser Effekt war sowohl beim GLVQ als auch bei dem GRLVQ für den Tecator-Datensatz zu erkennen. Für diesen Datensatz ist das Lernen des Parameters α ein entscheidendes Werkzeug. Hierfür wurde in der Abbildung 4.6 der enorme Leistungsanstieg in Abhängigkeit des Parameters α illustriert. Falls der Anstieg der Datenvektoren nicht ausschlaggebend für die Klassifikation der Daten ist, wird der Parameter α kleiner. Dieser Sachverhalt wurde für den Kaffee-Datensatz aufgezeigt. Außerdem kam es bei dem Kaffee-Datensatz sogar zu einer Konvergenz des Parameters. Diese brachte allerdings keine enorme Leistungssteigerung, wie bei dem Tecator-Datensatz. Der klassische GLVQ und der GLVQ mit der Sobolev-Metrik erzielten für den Kaffee-Datensatz die gleichen Ergebnisse. Die Sobolev-Metrik ist somit keine Verschlechterung für den GLVQ bzw. GRLVQ, sondern eine Erweiterung bezüglich der Anwendbarkeit des GLVQ bzw. GRLVQ.
Für den GRLVQ kam man zu der Erkenntnis, dass ein simultaner Lernprozess des Parameters α und des Relevanzvektors $\boldsymbol{\lambda}$ nicht die günstigste Wahl ist. Ein Grund dafür ist, dass für unterschiedliche Relevanzprofile bei den Anstiegen und bei den klassischen Distanzen der gleiche Vektor $\boldsymbol{\lambda}$ gelernt wurde (siehe Abbildung 5.4). Abhängig von dem Wert des Parameters α wird der Relevanzvektor in verschiedene Richtungen gelernt. Eine Möglichkeit Lösung des Problems ist die Anwendung von zwei Relevanzvektoren. Das brachte in den Tests zwar keine Leistungssteigerung, aber eine deutliche Darstellung beider Relevanzprofile. Auf Grund der Abhängigkeit der Relevanzprofile von α wurde auf einen separaten Lernprozess von α und $\boldsymbol{\lambda}$ hingewiesen.
Zum Anderen nutze man die funktionalen Eingenschaften der Datenvektoren dahingehend, dass ein Basissystem von Funktionen adaptiert wurde, die man zur Darstellung der Prototypen verwendete. Diesbezüglich beinhaltete das System Gaußfunktionen oder Sigmoidfunktionen. Man adaptierte somit die Prototypen nicht mehr hinsichtlich der einzelnen Dimensionen, sondern man lernte die Parameter der einzelnen Ba-

sisfunktionen. Zum Einen wurde dadurch die Anzahl der erlernten Parameter reduziert und zum Anderen konnten die Ableitungen der Prototypen analytisch berechnet werden und nicht mehr numerisch. Die Wahl der Basisfunktionen war stark von dem Verlauf der Datenvektoren abhängig. Demnach erkannte man, dass die Gaußfunktionen für den Tecator-Datensatz ungeeignet sind. Außerdem konnte gezeigt werden, dass die Anwendung von Sigmoidfunktionen eine größere Flexibilität in der Darstellung der Datenverläufe hervorbringt als es mit Gaußfunktionen möglich ist.
Bezüglich des GRLVQ bzw. GMLVQ wurde eine interessante Beobachtung für die Relevanzprofile bzw. Korrelationsmatrix gemacht. Die Verwendung von Basisfunktionen bewirkt eine Nachbarschaft zwischen den Datendimensionen. Das bedeutet: Wenn die Relevanz einer Datendimension steigt bzw. sinkt, dann steigen bzw. sinken auch die Relevanzen der Nachbardimensionen. Das hat zur Folge, dass die Relevanzprofile bzw. Korrelationsmatrizen geglättet werden. Damit kann mögliches Rauschen aus den Relevanzen entfernt werden. Allerdings können auch Informationen dabei verloren gehen. Prinzipiell ist jedoch die Anwendung von Basisfunktionen zur Darstellung von Prototypen kein Nachteil. Unter den funktionalen Eigenschaften der Datenvektoren ist es eine elegante Möglichkeit die Prototypen darzustellen.

Für die Zukunft könnten weitere Betrachtungen mit der Sobolev-Metrik gemacht werden. Beispielsweise könnte man die Ableitungen zweiter Ordnung in die Metrik mit eingehen lassen. Damit besteht zwar die Gefahr eines größeren Fehlers, da man die Ableitungen approximativ berechnet, aber es kann zusätzlich die Krümmung zur Klassifikation verwendet werden.
Bezüglich der Darstellung von Prototypen durch Basisfunktionen sollten mehrere Typen von Basisfunktionen getestet werden. Unter Anderem ist die Lorentzfunktion eine weitere Möglichkeit zur Darstellung von Prototypen. Hier kann auch ein Ansatz über verschiedene Typen von Basisfunktionen getestet werden. Anhand der Struktur der Datenvektoren kann vielleicht eine Verwendung von optimalen Basisfunktionen ermittelt werden. Unter optimale Basisfunktionen versteht man sehr gut geeignete Basisfunktionen, um die Verläufe der Datenvektoren zu repräsentieren.

Anhang A: Beweis Äquivalenz Sobolev-Norm

Man betrachte die Sobolev-Norm

$$\|\mathbf{x}\|_{S(p,R)} := \left(\|\mathbf{x}\|_{S(p,R)}\right)^p = \sum_{j=0}^{R} \|D^{(j)}\mathbf{x}\|^p_{\mathscr{L}^p(0,1)}, \tag{A.1}$$

wobei $D^{(j)}\mathbf{x}$ die j-te Ableitung der Funktion $\mathbf{x}$ und $\|\cdot\|_{\mathscr{L}^p(0,1)}$ die L_p-Norm über $[0,1]$ ist, d.h.

$$\|\mathbf{x}\|_{\mathscr{L}^p(0,1)} = \sqrt[p]{\int_0^1 |\,\mathbf{x}(t)\,|^p\,dt}.$$

Es soll nun gezeigt werden, dass (A.1) äquivalent ist zu dem Ausdruck

$$\|\mathbf{x}\|_{p,R} := (\|\mathbf{x}\|_{p,R})^p = \|\mathbf{x}\|^p_{\mathscr{L}^p(0,1)} + \|D^{(R)}\mathbf{x}\|^p_{\mathscr{L}^p(0,1)}.$$

Beweis: Zwei Normen $\|\cdot\|_A$ und $\|\cdot\|_B$ heißen äquivalent, wenn es zwei positive Konstanten c_1 und c_2 gibt, so dass gilt

$$c_1\|\cdot\|_B \leq \|\cdot\|_A \leq c_2\|\cdot\|_B \tag{A.2}$$

Nach der Definition für die Äquivalenz von zwei Normen folgt unmittelbar

$$\|\mathbf{x}\|_{p,R} \leq \|\mathbf{x}\|_{S(p,R)} \text{ für } c_1 = 1. \tag{A.3}$$

Nun bleibt zu zeigen, dass ein c_2 existiert, so dass gilt $\|\mathbf{x}\|_{S(p,R)} \leq c_2 \cdot \|\mathbf{x}\|_{p,R}$. Hierfür verwendet man den Hauptsatz der Differential- und Integralrechnung, d.h.

$$D^{(j)}\mathbf{x}^{(j)}(t) = \int_0^t D^{(j+1)}\mathbf{x}(c)dc \tag{A.4}$$

und die Minkowski-Ungleichung

$$\|\mathbf{x}+\mathbf{y}\|_{\mathscr{L}^p(0,1)} \leq \|\mathbf{x}\|_{\mathscr{L}^p(0,1)} + \|\mathbf{y}\|_{\mathscr{L}^p(0,1)}. \tag{A.5}$$

Wendet man die Gleichung (A.4) auf $\|D^{(k)}\mathbf{x}\|_p$ an, so erhält man

$$\|D^{(j)}\mathbf{x}\|_p = \left[\int_0^1 |\int_0^t D^{(j+1)}\mathbf{x}(c)dc\,|^p\,dt\right]^{\frac{1}{p}},$$

was wiederum mit der Dreiecksungleichung nach oben abgeschätzt werden kann, so

dass gilt

$$\left[\int_0^1 |\int_0^t D^{(j+1)}\mathbf{x}(c)\,dc|^p dt\right]^{\frac{1}{p}} \leq \left[\int_0^1 \int_0^t |D^{(j+1)}\mathbf{x}(c)|^p dcdt\right]^{\frac{1}{p}}.$$

Nach dem Vertauschen der Integrale ergibt sich

$$\left[\int_0^1 \int_0^t |D^{(j+1)}\mathbf{x}(c)|^p dcdt\right]^{\frac{1}{p}} = \left[\int_0^1 \int_c^1 |D^{(j+1)}\mathbf{x}(c)|^p dtdc\right]^{\frac{1}{p}}, \tag{A.6}$$

wobei auch die Integrationsgrenzen angepasst werden müssen. [7] Das rechte Integral kann nun ebenfalls nach oben abgeschätzt werden mit

$$\left[\int_0^1 \int_c^1 |D^{(j+1)}\mathbf{x}(c)|^p dtdc\right]^{\frac{1}{p}} \leq \left[\int_0^1 |D^{(j+1)}\mathbf{x}(c)|^p dc\right]^{\frac{1}{p}} = \|D^{(j+1)}\mathbf{x}\|_{\mathscr{L}^p(0,1)}.$$

Somit ergibt sich $\|D^{(j)}\mathbf{x}\|_{\mathscr{L}^p(0,1)} \leq \|D^{(j+1)}\mathbf{x}\|_{\mathscr{L}^p(0,1)}$. Weiter erhält man durch vollständige Induktion

$$\sum_{j=1}^{R} \|D^{(j)}\mathbf{x}\|_{\mathscr{L}^p(0,1)} \leq R \cdot \|D^{(R)}\mathbf{x}\|_{\mathscr{L}^p(0,1)}.$$

Jetzt kann folgende Ungleichungskette aufgestellt werden

$$\begin{aligned}
&||\sum_{j=0}^{R} D^{(j)}\mathbf{x}||_{\mathscr{L}^p(0,1)} \leq \sum_{j=0}^{R} ||D^{(j)}\mathbf{x}||_{\mathscr{L}^p(0,1)} \\
&\leq ||D^{(0)}\mathbf{x}||_{\mathscr{L}^p(0,1)} + R \cdot ||D^{(R)}\mathbf{x}||_{\mathscr{L}^p(0,1)} \leq R \cdot \left(||D^{(0)}\mathbf{x}||_{\mathscr{L}^p(0,1)} + ||D^{(R)}\mathbf{x}||_{\mathscr{L}^p(0,1)}\right).
\end{aligned}$$

Nach Anwendung der Minkowski-Ungleichung (A.5) erhält man die zu beweisende Aussage

$$||\mathbf{x}||_{S(p,R)} \leq R \cdot ||\mathbf{x}||_{p,R},$$

mit $c_2 = R$ in (A.2), was zusammen mit (A.3) zu beweisen war. □

[7] Man beachte für das linke Integral (A.6) den Integrationsbereich $M = \{0 < t < 1, 0 < c < t\}$ welcher äquivalent ist zu $M' = \{0 < c < t < 1\}$. Man trennt die letzte Bedingung zu $M^* = \{0 < c < 1, c < t < 1\}$, so dass c unabhängig ist von t aber t jetzt abhängig ist von c. Dies führt zu den Integrationsgrenzen, welche bei dem rechten Integral (A.6) gegeben sind.

Literaturverzeichnis

[1] Lumer, G.: *Semi-Inner-Product Spaces*, Transactions of the American Mathematical Society, Band 100, 1961

[2] Million, E.: *The Hadamard Product*, April 2007

[3] Villmann, T.; Hammer, B.: *Generalized Relevance Learning Vector Quantization*, Neural Networks, 2002

[4] Kästner, M.; Hammer, B.; Biehl, M.; Villmann, T.: *Generalized Functional Relevance Learning Vector Quantization*, n.A., 2009

[5] Riedel, M.; Nebel, D.: *Generalized Functional Matrix Learning Vector Quantization*, zugl. Masterthesis Hochschule Mittweida, 2011

[6] Fischer, L.: *Modifikation unüberwachter Vektorquantisierer für funktionale Daten und Einbindung einer neuen Optimierungsstrategie*, zugl. Masterthesis Hochschule Mittweida, 2012

[7] Kushner, H. ; Clark, D.: *Stochastic Appproximation Methods for Constrained and Unconstrained Systems*, Springer Verlag, 1978

[8] Pekalska, E.; Duin, R. P. W.: *The Dissimilarity Representation for Pattern Recognition*, World Scientific Publishing Co. Pte. Ltd., 2005

[9] Kreiß, J.-P.; Neuhaus, G.: *Einführung in die Zeitreihenanalyse*, Springer Verlag, 2006

[10] Holzer, S.: *Vorlesungsskript zur numerischen Differentiation und Differenzenverfahren*,
Online im Internet, URL:
`http://www.unibw.de/rz/dokumente/public/getFILE?fid=bs\_1419759`,
(Aufgerufen am: 21.03.2012)

[11] Saralajew, S.: *Learning Vector Quantization*, 2012

[12] Strickert, M.: *Enhancing [M|G]RLVQ by quasi step discriminatory functions using second order training*, Machine Learning Reports, 2011

[13] Biehl, M.; Schneider, P.; Hammer, B.: *Matrix Learning in Learning Vector Quantization*, Machine Learning Reports, 2008

[14] Sato, A.; Yamada, K.: *Generalized Learning Vector Quantization*, In: Advances in Neural Information Processing Systems 7 (NIPS), pp. 423 - 429, MIT Press, 1995

[15] Kohonen, T.: *The Self-Organizing Map*, In: Proceedings of the IEEE, Vol. 78, No. 9, pp. 1470 - 1472, 1990

[16] Kästner, M.; Hammer, B.; Biehl, M.; Villmann, T.: *Functional Relevance Learning in Generalized Learning Vector Quantization*, Neurocomputing, 2010

[17] Steinhardt, A.: *Untersuchung und Modellierung von Varianten des GLVQ-Algorithmus zur Klassifikation mehrdimensionaler Daten*, zugl. Bachelorthesis Hochschule Mittweida, 2010

[18] Haykin, S.: *Neural Networks and Learning Machines, 3rd ed.*, Pearson Prentice Hall, 2009

[19] Rossi, F.; Villa-Vialaneix, N.: *Consistency of Functional Learning Methods Based on Derivatives*, Pattern Recognition Letters, volume 32, number 8, pages 1197-1209, 2011

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 17.08.2012